RN-XCM: A Neural Network for Classifying Skill Level in Multi-Axis Tracking Tasks

MSc. Thesis

K.Z. Six

TU Delft

RN-XCM: A Neural Network for Classifying Skill Level in Multi-Axis Tracking Tasks

MSc. Thesis

Thesis report

by

K.Z. Six

to obtain the degree of Master of Science
at the Delft University of Technology
to be defended publicly on August 31, 2023 at 13:00

Thesis committee:

Chair:	Prof. dr. ir. M. Mulder
Supervisors:	Dr. ir. D. M. Pool
	Prof. dr. ir. M. Mulder
External examiners:	Dr. ir. W. van der Wal
	Dr. ir. M.M. van Paassen
Place:	Faculty of Aerospace Engineering, Delft
Project Duration:	November, 2022 - August, 2023
Student number:	4779045

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

Faculty of Aerospace Engineering · Delft University of Technology

Preface

This MSc thesis encapsulates the culmination of my aerospace engineering journey at TU Delft, marking my final work as a student. Comprising of three integral components, this thesis consists of (I) a scientific paper, (II) appendices to the scientific paper, and (III) a preliminary report consisting of a literature study and a preliminary analysis.

The past nine months dedicated to writing this thesis have been one of the most enriching and fulfilling experiences of my time as a student. Although I started this thesis with minimal exposure to the realm of Neural Networks, I willingly embraced the challenge, recognizing the immense potential they held. It is with immense gratitude that I extend my appreciation to my supervisors, Daan M. Pool and Max Mulder, for their unwavering support, their consistently positive and encouraging attitudes throughout the writing of this thesis, and their constructive and detailed feedback that helped push the quality of this thesis to a higher level.

Naturally, I would like to first and foremost express my sincerest gratitude to my parents and sister for not only supporting me through the writing of this thesis but also during my time at TU Delft. Your unwavering faith in me, despite the 6,000 km distance between us, truly supported me throughout the past 5 years and I hope I can continue making you all proud in the coming years. In addition, I would like to express gratitude to the people around me whilst living in the Netherlands who have filled my time here with cherished memories and irreplaceable joy.

Last but not least, thank *you*, the reader, for taking some time to read this work and I hope you enjoy it as much as I did to write it.

K.Z. Six
Delft, August 2023

Contents

Nomenclature

List of Abbreviations

$ALSTM\text{-}FCN$	Attention Long Short Term Memory - Fully Convolution Networks
CAM	Class Activation Mapping
CNN	Convolutional Neural Network
FNN	Feedforward Neural Network
GAP	Global Average Pooling
HO	Human Operator
IQR	Inter-Quartile Range
KBB	Knowledge-Based Behaviour
$LIME$	Local Interpretable Model-Agnostic Explanation
$LSTM$	Long Short Term Memory
$LSTM\text{-}FCN$	Long Short Term Memory - Fully Convolution Networks
MA	Moving Average
$MALSTM\text{-}FCN$	Multivariate Attention Long Short Term Memory - Fully Convolution Networks
MLP	Multilayer Perceptron
$MLSTM\text{-}FCN$	Multivariate Long Short Term Memory - Fully Convolution Networks
$MTEX\text{-}CNN$	Multivariate Time Series Explanations for Predictions with Convolutional Neural Networks
MTS	Multivariate Time Series
$MTSC$	Multivariate Time Series Classification
NN	Neural Network
OL	Overlap
RBB	Rule-Based Behaviour
$ReLU$	Rectified Linear Unit
$ResNet$	Residual Network
$RN\text{-}XCM$	ResNet-XCM
RNN	Recurrent Neural Network
SBB	Skill-Based Behaviour
SF	Sampling Frequency
$SHAP$	SHapley Additive exPlanation
$tanh$	Hyperbolic Tangent
$TICA$	Time-Invariant Condition Average
$TSEM$	Temporally-Weighted Spatiotemporal Explainable Neural Network for Multivariate Time Series
WS	Window Size
XAI	Explainable Artificial Intelligence
XCM	eXplainable CNN Method for MTS Classification

List of Symbols

$\dot{.}$	First-Order Derivative
θ	Pitch Attitude
ϕ	Roll Attitude
μ	Mean
σ	Standard Deviation
e	Tracking Error Signal
u	Pilot Control Output Signal
$\hat{u}$	Scaled Pilot Control Output Signal
$\hat{y}$	Predicted Output Value
ω_c	Crossover Frequency
ω_{nm}	Neuromuscular System Frequency
τ_m	Motion Dealys
τ_v	Visual Delays
ζ_{nm}	Neuromuscular Damping Ratio

a	Activation Value
b	Bias
C	Cell State
D	Number of Observed Input Variables
F	Number of Kernels within Resolution Block
f_t	Target Forcing Function
h	Hidden Layer
H_c	Controlled Element Dynamics
H_p	Human Operator Response
i	Input Layer
K	Number of Kernels outside Resolution Block
K_m	Motion Gain
K_s	Sidestick Gain
K_v	Visual Gain
L	Kernel Size
n	Remnant Signal
o	Output Layer
s	Laplace Operator
T	Length of Time Series
t	Time-step
T_I	Visual Lag Time Constant
T_L	Visual Lead Time Constant
w	Weight
x	Input Value
y	Actual Output Value

List of Figures

List of Tables

1

Introduction

According to Pool and Zaal [1], it is possible to assess a Human Operator's (HO) acquisition of manual control skills after training through the use of a Cybernetic approach. In other words, they were able to use Cybernetics to assess pilot skill levels. However, the use of such an approach requires the need for certain assumptions. In the use of Cybernetics, Pool and Zaal [1] assume a quasi-linear pilot model which results in a linear describing function of the HO that applies only to a compensatory display. In doing so, the researchers could not extract any information regarding the non-linear behaviour of the HO, which is critical in understanding the HO control strategy [2]. This is for the fact that human control behaviour is known to be time-varying, non-linear, stochastic, and intermittent which cannot be captured by any state-of-the-art Cybernetic approach. In addition, the use of the Cybernetic approach requires obtaining a significant amount of lengthy time recordings of HO control behaviour to be accurate. This removes any possibility for real-time HO skill classification which could help increase the synergy between pilots and the aircraft they are in control of. With all of this in mind, Deep Learning, the use of artificial Neural Networks (NN) to perform tasks, is proposed as a suitable candidate to overcome the shortcomings of using the Cybernetic approach in assessing the non-linear control strategy of HOs with the possibility of real-time usage.

Deep Learning is proposed in the use of non-linear information extraction over other Machine Learning methods for a couple of key factors. The first key factor is Deep Learning's ability to show a continued increase in performance with an increased amount of data sets compared to other Machine Learning methods which when put in similar situations, hit a performance plateau. Deep Learning's ability to do so makes it extremely viable for this specific research as humans show variance in their control strategy when compared to one another and therefore necessitates the use of large data sets of HO control behaviour to accommodate these variances [3]. The second key factor is the need for feature extraction by the user when using other Machine Learning methods [3]. As the goal of this research is a model that would be able to identify and extract this non-linear information, the user should not be tasked to do so. Deep Learning circumvents this and learns to identify these features themselves without the need for any feature extraction by the users.

The main goal of this research is to produce a Deep Learning model which would be able to classify a HO's skill level based on the multivariate time series (MTS) data that is their control strategy of a two-axis tracking task. Success in creating such a multivariate time series classification (MTSC) model would allow for advancements in human-machine interactions as a whole and not only within the realm of aviation. Example uses of such a model would include, but are not limited to, a system that would allow for scalable levels of autonomy in real-time or a system that would alert a pilot if it detects a decrease in skill level, requiring the pilot to reassess themselves in real-time to ensure a high level of performance.

This thesis is structured as follows. Part I is the scientific article that presents the main findings of this research. Part II presents a preliminary report that was performed to help guide the structure and scope of the research. Part III presents the appendices of the scientific article.

Part I

Scientific Article

RN-XCM: A Neural Network for Classifying Skill Level in Multi-Axis Tracking Tasks

Kobi Z. Six, *Author*, Daan M. Pool, *Supervisor*, and Max Mulder, *Supervisor*

***Abstract*—The aviation industry's reliance on automation raises concerns about pilot complacency, necessitating continuous pilot proficiency measures. To that end, real-time pilot skill feedback is vital—through alerts on declining skill levels or scalable levels of autonomy. Current cybernetic methods are limited as they assume linearity and time-invariance of human behavior and lack real-time capability. Neural Networks (NNs) offer a solution but face challenges such as high computational costs and limited generalization capability. To overcome these issues, this paper introduces a new and compact Residual Network for eXplainable Convolutional MTS Classification (RN-XCM) designed explicitly to classify pilot skill levels. Results demonstrate RN-XCM's ability to accurately classify skill levels based on 1.2 seconds of dual-axis control data, achieving a test accuracy of up to 93.50%, while requiring 50% less training time than competing NN models. It also achieves a test accuracy of 80.16% for previously unseen subjects, signifying its competence as a one-size-fits-all classifier. Notably, RN-XCM performs 17.88% better when classifying dual-axis tracking tasks over single-axis tracking tasks. Overall, the possibility of real-time feedback provided by the RN-XCM can enable quantitative evaluation of pilot control behavior, therefore enhancing safety and facilitating smoother interactions between pilots and aircraft.**

***Index Terms*—Manual control, time series classification, multivariate time series classification, deep learning, neural networks, skill level**

NOMENCLATURE

Acronyms / Abbreviations

$1D$	1-Dimensional
$2D$	2-Dimensional
BN	Batch Normalization
CAM	Class Activation Mapping
CNN	Convolutional Neural Network
CV	Cross-Validation
DL	Deep Learning
FM	Feature Map
GAP	Global Average Pooling
$HMILab$	Human-Machine Interaction Laboratory
HO	Human Operator
LW	Label Width
ML	Machine Learning
MTS	Multivariate Time Series
NN	Neural Network
OL	Overlap
$ReLU$	Rectified Linear Unit
$ResNet$	Residual Network
$RN\text{-}XCM$	ResNet-XCM
RNN	Recurrent Neural Network
SF	Sampling Frequency
SRS	SIMONA Research Simulator
TS	Time Series
$TU\ Delft$	Delft University of Technology
UTS	Univariate Time Series
WS	Window Size
XCM	eXplainable CNN Method for MTS Classification

Symbols

$\dot{}$	First-Order Derivative
θ	Pitch Attitude
ϕ	Roll Attitude
μ	Mean
σ	Standard Deviation
e	Tracking Error Signal
u	Pilot Control Output Signal
$\hat{u}$	Scaled Pilot Control Output Signal
D	Number of Observed Input Variables
F	Number of Kernels
f_t	Target Forcing Function
H_c	Controlled Element Dynamics
K	Kernel Size
K_s	Sidestick Gain
n	Remnant Signal
s	Laplace Operator
T	Length of Time Series
u_c	Crossfeed Pilot Control Output Signal

I. INTRODUCTION

In recent years, automation has become an integral aspect of most manual control tasks such as driving cars and piloting aircraft. However, this increase in automation has raised concerns regarding safety as it essentially changes the human role to passive monitoring rather than active participation [1]. This change brings with it potential drawbacks such as complacency, reduced awareness, and overall skill degradation [2]–[5]. With pilots being responsible for the lives of both the passengers and the crew members of the aircraft they are operating, this degradation or loss of skill is clearly undesired. Therefore, it has become increasingly important to ensure that pilots possess the necessary skills and expertise to operate an aircraft safely and effectively. Consequently, a real-time quantitative measure of a pilot's manual control skill would allow for safer and more effective operation through possible adaptive automation and decision support.

Although quantitatively assessing the skill level of a Human Operator (HO) is not easily done, the use of cybernetics has

been shown to have the ability to do so [6]–[8]. However, traditional cybernetic methods require the assumption that the HO is constant and linear in their control actions over time. This assumption neglects the inherent time-varying nature of humans and therefore does not capture any of the subtle, but possibly important changes in their behavior over time [9], [10]. Though there have been efforts in modeling this non-linear and time-varying behavior within cybernetics [11]–[14], obtaining information regarding HO control behavior from such fundamentally noise-ridden data remains extremely difficult [9]. In addition, to obtain meaningful results, traditional cybernetic methods require long data sequences of HO task runs, removing any possibility of real-time classification [6].

To capture the non-linear behavior of HOs, Machine Learning (ML) has been used in place of cybernetics [15]–[18]. This use of ML to capture non-linear behavior, however, required at least 60 s of time-series data to provide accurate results [18]. This time range prevents the possibility of real-time classification. In addition, the use of ML necessitated manual input feature engineering alongside the need for additional sensors such as face cameras, eye-tracking devices, and electrocardiograms which makes implementation impractical [15], [16], [18].

To allow for both the capturing of the non-linear behavior of HOs whilst also allowing for real-time classification of skill without the need for either feature engineering or additional sensors, the use of Deep Learning (DL), a subset of ML that uses Neural Network (NN) models, has been explored [19]–[21]. NNs have been shown to obtain exceptional classification accuracies in not only Univariate Time Series (UTS) classification [22] but also Multivariate Time Series (MTS) classification [23]. As the signals obtained regarding HO control behavior are MTS signals, DL has shown great promise in classifying them [19], [20]. Most relevantly, NNs have been used to classify the skill level of HOs based on their control behavior in single-axis tracking tasks with promising validation accuracies up to 92% [21]. Therefore, this approach will be expanded upon to classify the skill level of HOs in dual-axis tracking tasks, which more closely mimics the tasks performed by pilots when operating a real aircraft.

The use of NNs in multi-axis tracking tasks presents a specific challenge characterized by the emergence of four distinct phenomena. These phenomena stem from the existence of multiple axes and have the potential to significantly impact the classification performance of the NN. They include *Performance Degradation*, *Axis Asymmetry*, *Crossfeed*, and *Intermittency* [24].

This paper proposes a new Convolutional Neural Network (CNN) model for MTS classification which will be used to produce a novel pilot skill-level classification method in dual-axis tracking tasks that uses readily available task-related time series signals as input features. This NN model was developed as current MTS NN classifiers are computationally expensive and have low generalization ability [25]. Overall, this NN model overcame the single-dimension analyzing methods found in most Time Series (TS) classification models to improve the overall performance of the model while lowering the required computational workload compared to most current MTS classification models. This NN model was trained to classify the time-series data as "skilled" or "unskilled" based on the time-series signals obtained from a previous dual-axis tracking task experiment [26] without the need for any additional and external sensors. In this paper, the model was trained, validated, and tested on experimentally obtained human control data from previously conducted tracking tasks with the model obtaining strong test accuracies on control data for dual-axis tracking tasks.

The paper is structured as follows. Section II presents background information on dual-axis tracking tasks, the three datasets used, and the data pre-processing method. Section III presents background information on NN TS classifiers, the RN-XCM model, and the model's settings. Section IV details the steps used to optimize and evaluate the model. Section V presents the results and Section VI discusses said results. Section VII concludes the paper.

II. Datasets and Data Handling

A. Background: Dual-Axis Compensatory Tracking Task

The closed-loop manual control task aimed to mimic the operation of an aircraft is a dual-axis compensatory tracking task in which the HO simultaneously controls both the pitch, θ, and roll, ϕ, attitude [24], [26]. This differs from most research involving manual control tasks as these tend to be limited to single-axis control tasks [9]. Those that do tackle multi-axis control tasks tend to do so by modeling them as if they were entirely independent single axes [27]–[29]. This, however, is an incomplete view as this separation of axes no longer takes into account the inter-axes relationships that occur with these multi-axis tracking tasks as research has shown that multi-axis tracking tasks are distinct from single-axis tracking tasks [30]–[33].

A schematic representation of a compensatory dual-axis tracking task is shown in Fig. 1. In the task, the HO responds only to visual cues, i.e. the pitch and roll errors, e_θ and e_ϕ, through a compensatory display, resulting in the pitch and roll human control output, u_θ and u_ϕ. These control outputs are then fed into the controlled element, the aircraft, which has pitch and roll dynamics, $H_{c,\theta}$ and $H_{c,\phi}$, though the use of a sidestick with pitch and roll gains of $K_{s,\theta}$ and $K_{s,\phi}$. As the name suggests, the goal of the HO is to control the pitch and roll attitudes, θ and ϕ, in order to track the pitch and roll forcing function $f_{t,\theta}$ and $f_{t,\phi}$ as closely as possible to minimize the error shown in the compensatory display. It can be seen the human control output in each axis u is a summation of three signals: the remnant signal that accounts for nonlinear behavior and measurement noise [34], n, the human control output as a response to principal-axis error signals, u_e, and the human control output as a response to the off-axis error signals, i.e. crossfeed [24], u_c.

For dual-axis tracking tasks, the presence of four distinct phenomena that may occur due to the addition of another axis was previously reported [24]:

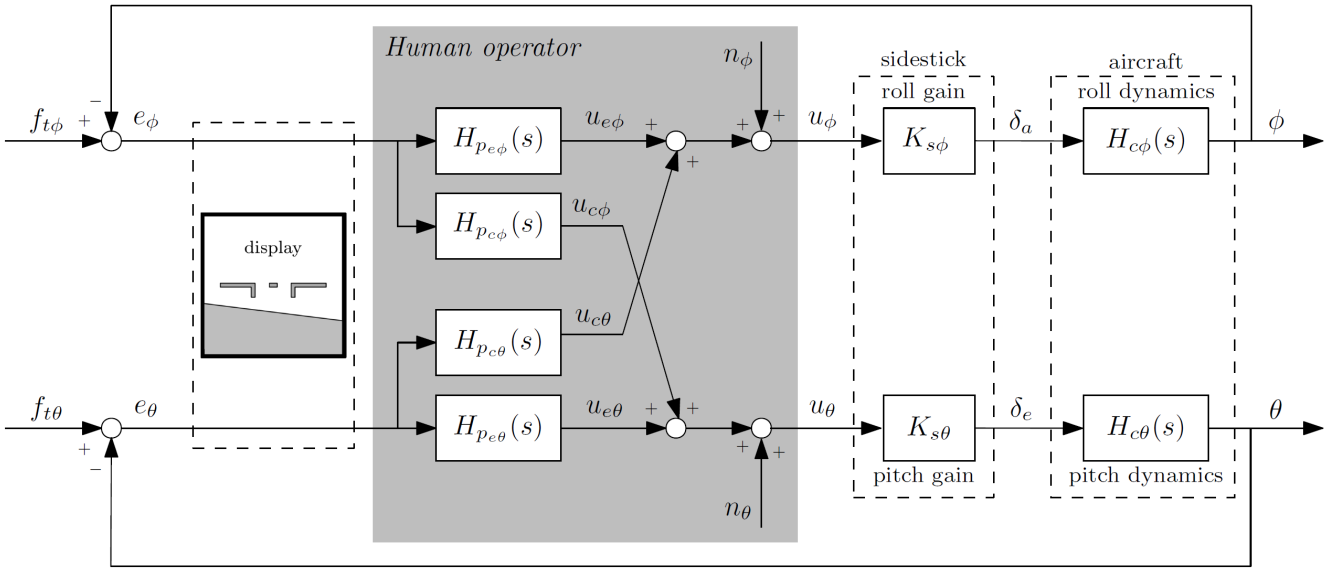

Figure 1: Schematic representation of a compensatory dual-axis pitch and roll tracking task with crossfeed present.

- *Performance Degradation*: The observed degradation of performance in dual-axis tasks when compared to single-axis tasks.
- *Axis Asymmetry*: The distinct difference in the performance of each axis.
- *Crossfeed*: A form of task interference in which HOs are unable to decouple the tasks needed per axis.
- *Intermittency*: The time-varying axis prioritization of human operators in dual-axis control tasks, i.e., attention switching.

In terms of skill acquisition within dual-axis tracking tasks, it was discovered that a number of the aforementioned phenomena were more noticeable than others. Reference [26] found that for an aircraft's pitch-role tracking task, the tracking error was consistently lower in the pitch axis than in the roll axis, which is in line with the findings of [24] regarding *Axis Asymmetry*. This phenomenon, however, was not entirely limited to per-axis performance as it was also found that participants displayed both higher learning rates and earlier stabilization during training of the task in the pitch axis. In addition, although [24] stated the importance of *Crossfeed* regarding HO control inputs during the task, up to 20% contribution of HO control inputs, [26] found that in terms of skill acquisition, this was not the case. They discovered that *Crossfeed*'s contribution was only up to 8% and was constant even with an increase in skill. Rather, it was discovered that the remnant signal, the non-linear aspect of the HO control behavior, had far greater contributions of up to 80%. The phenomenon of *Axis Asymmetry* was again present where it was found that the decrease of remnant contribution was greater in the pitch axis than in the roll axis, confirming the preference in the pitch axis regarding skill acquisition. In addition, it was found that these remnant contributions were 25% higher when the HOs were untrained, i.e. early in the training runs, compared to when they were trained, i.e. later in the training runs [26].

B. The Three Datasets

To be able to assess the ability of the RN-XCM model to classify the skill level of HOs in tracking tasks, datasets from three different experiments will be used [8], [24], [26].

1) Experiment 1 (RW Data): The experiment by Wijlens et al. [26] was conducted in order to objectively and quantitatively evaluate the acquisition, decay, and retention of skill in compensatory dual-axis tracking tasks. The experiment was performed in the Human-Machine Interaction Laboratory (HMILab) at Delft University of Technology (TU Delft). Within the experiment, 38 task-naive participants had to undergo both a training phase and an evaluation phase. As the current study is concerned solely with the acquisition of skill, only the training data will be used for the rest of the study. In the training phase, the participants underwent 100 runs split into four days with each run lasting 90 s of which the last 81.92 s were used as measurement data and was measured with a sampling frequency of 100 Hz.

Different pitch and roll dynamics, $H_{c,\theta}(s)$ and $H_{c,\phi}(s)$ respectively, were used to ensure that the control task is as close to simulating the actual feeling of controlling an aircraft as possible. The dynamics used were those of a medium-sized twin-engine transport aircraft, similar in size to a Boeing 757 with a gross weight of 185,800 lbs [26]. The linearised pitch and roll dynamics are formalized as (1) and (2) respectively with the dynamics linearised at a flight condition close to the stall point, at an airspeed of 150 kts and an altitude of 41,000 ft [26].

$$H_{c,\theta}(s) = \frac{0.333\left(s^2+0.092s+0.003\right)}{(s^2-0.014s+0.004)(s^2+0.446s+0.475)} \quad (1)$$

$$H_{c,\phi}(s) = \frac{0.768\left(s^2+0.220s+0.593\right)}{(s+0.736)(s-0.020)(s^2+0.146s+0.660)} \quad (2)$$

2) Experiment 2 (MJ Data): The dataset used by De Jong in their research [21] was obtained from an experiment conducted by Pool et al. [8]. The experiment aimed to determine the effects simulator motion feedback has on the acquisition of skill for a compensatory single-axis tracking task. In addition, the experiment was performed in the SIMONA Research Simulator (SRS) at TU Delft. Within the experiment, two groups of 12 task-naive participants had to undergo both a training phase and an evaluation phase. Similar to what was done for the RW Data, as the current study is concerned solely with the acquisition of skill, only the data of the training runs will be used for the study. In addition, as the RW Data was obtained from an experiment with no simulator motion feedback, only the data from the group that underwent a training phase with no motion feedback will be used to allow for a fair comparison of performances. Similar to the training phase of the RW Data, the participants underwent 100 runs split into four days where each run lasted 90 s of which the last 81.92 s were used as measurement data and was measured with a sampling frequency of 100 Hz.

The controlled element dynamics, $H_{c,\theta}(s)$, used for the single-axis compensatory tracking task was a reduced-order linearized model for the elevator-to-pitch dynamics of a Cessna Citation I and is formalized as (3) [35].

$$H_{c,\theta}(s) = 10.62\frac{s+0.99}{s\left(s^2+2.58s+7.61\right)} \quad (3)$$

3) Experiment 3 (SB Data): The experiment conducted by Barendswaard et al. [24] aimed to study the four distinct phenomena that occur due to multi-axis control tasks: *Performance Degradation*, *Axis Asymmetry*, *Crossfeed*, and *Intermittency*. Similar to the MJ Data, the experiment was conducted in SRS at TU Delft. For the experiment, 12 participants performed both dual-axis and single-axis tracking tasks with and without simulation motion feedback. For the same reasons as the MJ data, only the data with no simulation motion feedback will be used in the current study. Unlike both the RW Data and MJ Data, this experiment was not a training experiment, therefore the training phase was not as extensive and the respective training runs were not recorded. Hence, only the full evaluation runs will be utilized. As was done for the other experiments, each measured run lasted 90 s of which the last 81.92 s were used as measurement data and were measured with a sampling frequency of 100 Hz.

The controlled element dynamics used for both the single-axis and dual-axis compensatory tracking task is formalized as (4) [24]. Note that for the dual-axis task, this meant that both the pitch and roll dynamics were set to (4).

$$H_{c,\theta}(s) = H_{c,\phi}(s) = \frac{67.9}{s(s+3)} \quad (4)$$

C. Data Pre-Processing

Before any of the datasets could be used by the NN model, a few pre-processing steps had to be taken to prepare them. The goal of pre-processing the data is to improve the overall quality of the data and to prepare them for training the NN model, thus optimizing the model's performance in the task at hand. The data pre-processing steps taken in this research will be those taken by De Jong [21]. This was done to maintain consistency with the prior research and ensure successful outcomes as the methodology was employed not only by De Jong but also by Versteeg [19] and Verkerk [20]. In addition, the hyperparameters optimized for each data pre-processing step obtained by De Jong [21] will be adopted in this research. This was done as De Jong optimized these hyperparameters for the same task, classification of skill level, and will allow for significant time savings. The data pre-processing steps that will be taken in order are listed below.

1) Data Labeling: In order for the NN model to be able to make accurate predictions, a supervised learning approach will be utilized. This approach however requires the data to be labeled as the labels serve as the supervisor for the model, essentially dictating whether or not the model's output is correct. Therefore, all the aforementioned datasets have to be labeled such that a singular tracking run will either be classified as either "skilled" or "unskilled".

Given the goal of De Jong's study, and to an extent this study, is the classification of skill level based on control behavior, labeling the skill level of the tracking runs based on experience was deemed to be most appropriate [21]. This was done as this approach allows for the unbiased definition of skill level as undoubtedly, the first runs of a HO undergoing the task will be "unskilled" and the last runs will be "skilled". The use of this labeling method introduces the hyperparameter Labeling Width (*LW*). This hyperparameter, which De Jong has optimized [21], dictates the first and last number of runs of a singular HO would be labeled "unskilled" and "skilled", respectively, and in addition, allows for the symmetric labeling of runs which prevents class imbalance and bias in the NN model [36], [37]. Note that this labeling process is applicable only to both the RW Data and MJ Data as they originated from training experiments and thus had training runs to label. Conversely, SB Data originated from a non-training experiment and therefore had no training runs to label. Therefore, their entire evaluation runs would be used and would therefore be all labeled as "skilled".

2) Data Scaling: Once the data has been labeled, the next step in the pre-processing step is to scale the data. This step is done for two major reasons. The first is to allow the model's performance to converge faster to the highest classification accuracy [38]. The second reason is to ensure that by standardizing the data into a common scale, the model would actively learn the relationship between the variables with respect to time rather than looking at feature magnitude to classify skill level [39].

The data scaling method utilized by De Jong [21] was standardization along each tracking run. An example of the

standardization formula when applied to the u signal is shown in (5). Through the use of this formula, the mean μ is subtracted from the sequence and is then scaled along the entire sequence to the unit variance σ^2. This effectively causes the mean to equal to zero, μ = 0, and the standard deviation to equal to 1, σ = 1. De Jong utilized this method as Versteeg [19] discovered that it allowed for extremely high performance whilst maintaining the possibility of real-time classification.

$$\hat{u} = \frac{u - \mu}{\sigma} \tag{5}$$

3) Data Partitioning: The next step in the pre-processing of the data came in the form of partitioning the data, i.e., the sectioning of the data. This was done to increase the number of data available and time samples within each dataset for the training, validating, and testing of the RN-XCM model such that the performance increases. The data partitioning step contains three aspects that are carried out simultaneously with each being a hyperparameter that De Jong has optimized [21]. They are as follows:

- Window Size (*WS*): This hyperparameter, measured in s, refers to the sectioning of a HO's entire tracking run to create more samples. For example, if the tracking run is 100 s and the *WS* is 10 s, the tracking run will yield 10 non-overlapping samples for the model to train on. *WS* is utilized for two major reasons. The first reason is to exploit NN's ability to increase its classification performance with an increasing amount of data to train on [40]. The second reason is to allow for real-time classification. For example, if the model was trained on the full 100-second tracking run, then any future application of the model would either require the need for the entire tracking run to be completed beforehand, which does not allow for any real-time feedback to the human controller, or the need for the HO to wait for 100 s before getting any feedback. This feedback would not be meaningful to the HO as its prediction would be based on long-term behavior rather than their current short-term behavior. Note that this hyperparameter is separate from the similarly named *Window Size* hyperparameter found within the RN-XCM model and to ensure no confusion will occur, the one referring to data partitioning will be abbreviated and referred to as *WS*.
- Sampling Frequency (*SF*): This hyperparameter, measured in Hz, determines the number of samples that is available per second of time series data. For example, if the tracking run was recorded at 100 Hz, the tracking run would then contain 100 values per second. For an *SF* of 50 Hz, the number of values contained per second would be reduced to 50. This reduction of data per sample was done as it was discovered that long sequences of time series data resulted in degradation of performance in NN models [41].
- Overlap (*OL*): This hyperparameter, measured in %, is an extension of the *WS* parameter and determines the amount of overlap each "window" has with its neighboring "window". For example, if an overlap percentage of 50% is used, then the last 50% of data found in window n would be the exact same as the first 50% of data found in window $n+1$. Similar to the use of *WS*, as the use of *OL* would allow for the creation of more samples from a single run, it is used to exploit NN's ability to increase its classification performance with an increasing amount of data to train on [40] and in addition, an increase in consistency within the training data.

4) Input Variable Selection: The last step in the pre-processing of the data was to determine which variables of the input time trace would yield the highest classification performance. This step is important as the use of certain variables could result in either a positive or negative impact on the model's classification performance. For example, although more variables would theoretically allow for better results from the model, this could also allow for overfitting of the model [42]. In addition, certain variables could interact with one another such that the model would create "trivial" classifications based on the relation of these variables rather than the adaptation of the human control behavior [19]. Therefore, De Jong optimized the combination variables which would allow for the most ideal performance [21].

A summary of the hyperparameters alongside the values De Jong obtained after conducting an extensive hyperparameter optimization search that will be used in the current study is shown in Table I. Note that regarding the input variables, the variables used by De Jong, $e+\dot{e}+u+\dot{u}$, are applicable only to one-axis tracking tasks and therefore, to be applicable for the current research focus on dual-axis tracking tasks, the variables used by De Jong will be used for both axes of the tracking tasks, pitch, θ, and roll, ϕ.

Table I: The optimal data pre-processing hyperparameters and methods as optimized in [21]

Hyperparameter	De Jong's Optimal Values or Methods
LW	20
Scaling Methods	Standardizing Entire Tracking Run
SF, Hz	50
WS, s	1.2
OL, %	90
Input Variables	$e_\theta + \dot{e}_\theta + e_\phi + \dot{e}_\phi + u_\theta + \dot{u}_\theta + u_\phi + \dot{u}_\phi$

III. Neural Network

A. Background: Multivariate Time Series Neural Network Classifiers

When looking at the majority of NN models used for MTS classification tasks, one NN archetype is used most prevalently, CNNs [22], [23], [43]–[47]. Although other NN archetypes are used, such as the Recurrent Neural Networks (RNNs) used in [44] and [47], they are generally used alongside CNNs rather than independently. The classification problem being considered is shown in Fig. 2 where the NN will be used as a classifier that takes a small window of time-domain signal samples as input and outputs the classifier's

predicted probabilities of the observed window being "skilled" or "unskilled".

Preliminary analysis [25] found that for the classification of the skill level of dual-axis control behavior, NN models that analyze the control behavior with respect only to time [22], [44], [48] showcased exceptional generalization capabilities but were computationally expensive. Out of these NN models, ResNet [22], [48] performed the best obtaining a test accuracy of 80.92%. In contrast, NN models that analyze the control behavior with respect to both the input variables and time [45]–[47] showcased mediocre generalization capabilities but were more computationally efficient, requiring on average 98.39% less training time. These models came in two forms: those that analyze both dimensions in series [45] and those that analyze them in parallel [46], [47]. It was reported by [46] that the former suffered poor performance as it does not extract information relative to both the input variables and timestamps directly from the input data. Therefore, out of the NN models that analyze the input variables and timestamps in parallel, XCM [46] performed the best at 66.88%.

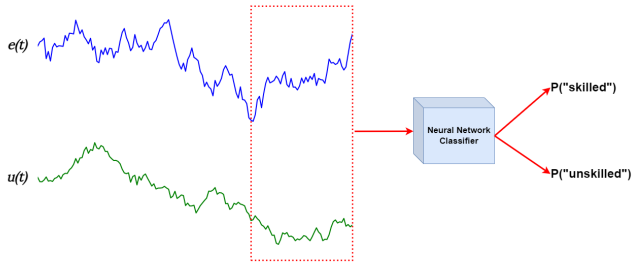

Figure 2: Schematic representation of the pilot skill level classification task using a NN classifier, adapted from [49].

1) Convolutional Neural Networks: CNNs are NNs that mimic the functionality of the visual cortex, the part of the brain responsible for processing visual information [42], [50], in which it hierarchically processes visual information through stages where each one builds on the features learned from the previous stage. CNNs mimic this through the use of *convolutional layers*. Each convolutional layer uses kernels, also referred to as filters, that essentially slide across the given TS data to produce Feature Maps (FMs) by applying its weights and biases to each snippet of the TS data [51]. Note that each kernel produces a singular FM. These FMs are then used as input for the subsequent convolutional layer such that earlier layers recognize the lower-level features and the later layers recognize the higher-level features, mimicking the visual cortex.

Concerning the NN model introduced in this study, it is important to acknowledge the existence of two different types of convolutional layers and the characteristic that distinguishes them, 1-Dimensional (1D) convolutional layers and 2-Dimensional (2D) convolutional layers. The characteristic that distinguishes them is the direction in which the kernel slides across the given data. Specifically, the former has its kernel slide in only one direction whereas the latter slides in two directions. Note that a convolutional layer can have a 2D kernel and still be a 1D convolutional layer as long as said kernel still travels in only one direction [52].

2) ResNet: Residual Network (ResNet) [48] is a 1D CNN model that takes advantage of the increase in performance found with an increase in the depth of the CNN, i.e., a greater number of convolutional layers in series within the NN [53]. ResNet has found great success in not only object detection and image classification tasks [48], but also in TS classification tasks [21], [22]. However, the increase in convolutional layers in series found in an NN introduces and exemplifies the vanishing gradient problem [54], in which the gradients used to update the weights of the NN model diminish to an extent that it becomes negligible, rendering it useless for any meaningful learning. To combat this, ResNet employs the use of *shortcut connections* between layers to allow for direct gradient flow.

3) XCM: eXplainable CNN method for MTS classification (XCM) [46] is a NN model that analyses information relative to both the signal value and time dimensions of the MTS data, through the use of both 1D and 2D convolutional layers. This contrasts ResNet's ability to only extract information relative to time due to its usage of solely 1D convolutional layers. The extraction of information relative to both the input variables, i.e. the e and u signals, and time enables improved generalization ability. This involves identifying which variables during which time segments were most responsible and which time segments were the contribution of all the input variables most responsible for a given skill classification.

B. The RN-XCM Model

RN-XCM achieves its goal of combining the best of both ResNet and XCM, great generalization ability and low computational workload respectively [25], by first using the structure of XCM as the foundation of the model and replacing every convolutional layer that analyses along the time dimension with a residual block as used by ResNet [22]. This was done due to the fact the original usage of the ResNet model for TS classification was applied only along the temporal dimension due to the usage of only 1D convolutional layers [22]. The use of XCM's network architecture as the basis for RN-XCM allowed for 96.36% shorter training times compared to ResNet [25]. The overall architecture can be seen in Fig. 3.

The RN-XCM model extracts information relative to the variables of the MTS data with 2D convolutional layers. This upper path of the model begins with a singular 2D convolutional layer followed by a Batch Normalization (*BN*) layer, to enable faster convergence in performance and allow for better generalization capabilities [55], and a Rectified Linear Unit (*ReLU*) layer, to induce non-linearity [56]. This 2D convolutional layer will extract information using a sliding kernel of stride 1 to produce F amount of FMs of dimension $D \times T$ through the use of padding, therefore maintaining the dimensions of the input MTS data. This use of padding is used to avoid upsampling the FMs in regards to the dimension from which information has been extracted. In order to determine

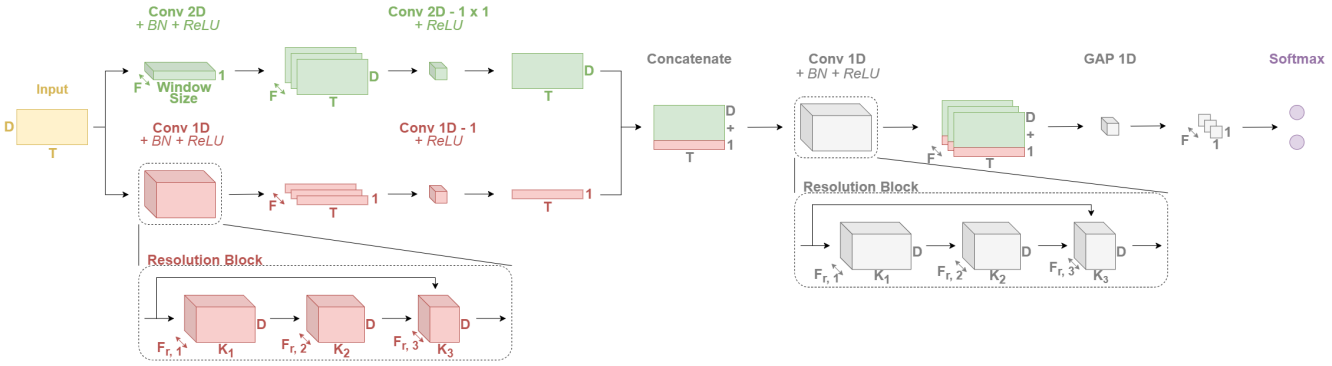

Figure 3: Model architecture of RN-XCM. Abbreviations: *BN*—Batch Normalization, *ReLU*—Rectified Linear Unit, *GAP*—Global Average Pooling, *Window Size*—kernel size which corresponds to a percentage of T, K—kernel size which is predetermined as per ResNet [22], D—number of observed input variables, T—time series length, F—number of kernels, and $F_{r,\,n}$—the number of kernels in the n-th convolutional layer of the resolution block

the kernel size of the 2D convolutional layer, RN-XCM incorporates the unique model hyperparameter *Window Size* which is defined as a percentage of the time length of the MTS input data and necessitates optimization [46]. The resulting FMs after this 2D convolution layer are then put through another 2D convolutional layer of kernel size 1×1 to reduce the number of FMs from F to 1, reducing the number of parameters [57], through channel-wise pooling followed by a *ReLU* layer.

In parallel, the model extracts information relative to the temporal dimension of the MTS data with 1D convolution layers. This path begins a residual block consisting of 1D convolutional layers that slide over the time axis in strides of 1 to capture the interactions between all the variables and produce F amount of FMs of dimension $1 \times T$ through the use of padding, which is done for same reasons as for the variable analyzing path of the model. Note that $F_{r,\,3}$ has to equal F in order to successfully produce F amount of FMs. The kernel sizes of the convolutional layers within the residual block are predetermined as per [22]. These FMs are then reduced to a singular FM of the same dimension through the use of a singular 1D convolutional layer of kernel size 1 to reduce the number of parameters [57] followed by a *ReLU* layer.

Once FMs have been produced for both dimensions, they are then concatenated to produce a singular FM of $(D + 1) \times T$. The same residual block consisting of 1D convolutional layers used in the temporal path of the model is then applied to this new FM where it slides over the time axis to capture the interactions between the features extracted from both dimensions. The resulting F features maps of dimension $1 \times T$ are then put through a 1D Global Average Pooling (GAP) layer with classification finally done with a softmax layer. The use of the GAP layer is to improve the generalization ability of the network in contrast to that shown through the use of fully connected layers [58].

One characteristic to note regarding the RN-XCN model that ensures its low computational costs is the fact that the size of the model is not dependent on the number of input variables. For example, if the model was used for the skill classification of single-axis tracking tasks and then for dual-axis tracking tasks, implying an increase in input variables by a magnitude of two, the model would not increase in size by that same magnitude. This is achieved as the number of convolutional layers and the number of FMs produced by each of these layers are set to a fixed amount as is done in the majority of NNs built for TS classifications [22], [23], [43]–[47].

C. Model Settings

Similar to the pre-processing steps, NN models also contain hyperparameters that can be tuned to allow for better performance. As RN-XCM is based on already existing models, there are certain hyperparameters that are unique only to those models, i.e., model-specific hyperparameters. Alongside these, there also exist model-agnostic hyperparameters, parameters that exist regardless of the models RN-XCM was based on.

1) Model-Specific Hyperparameters: Regarding the model-specific hyperparameters, it is important to note that due to the fact that RN-XCM is based on two different NN models, there exist two types of model-specific hyperparameters, those associated with ResNet [22] and those associated with XCM [46]. The values used for these model-specific hyperparameters will be those found in their corresponding research papers.

- *ResNet*: The hyperparameters corresponding to ResNet are those found within the resolution blocks of the RN-XCM model. To be specific, they are d, the depth, or the number of convolutional layers, of each resolution block, K, the kernel sizes for each convolutional layer in the resolution block, and the ratio of F_r, the ratio of the number of kernels in each convolutional layer in the resolution block relative to the first convolutional layer. As obtained from [22], the hyperparameter values are shown in Table II.
- *XCM*: The hyperparameters corresponding to XCM are those found outside the resolution blocks of the RN-XCM model. There exists only one hyperparameter and it is

Window Size, the kernel size of the singular convolutional layer expressed as the percentage of the time series length. However, [46] never obtained a general optimized value for *Window Size* and instead optimized it for each unique classification task they used to test the model. Therefore, this hyperparameter will have to be optimized for the task of classifying skill levels. The values of this hyperparameter that will be tested are those that were tested in [46]: [20, 40, 60, 80, 100]. As was stated before, this hyperparameter is independent of the similarly named hyperparameter used in the data partitioning step, *WS*.

Table II: The hyperparameter values for the resolution blocks as obtained in [22]

Hyperparameter	Values
d	3
K_1, K_2, K_3,	[8, 5, 3]
Ratio of $F_{r,\ 1}$, $F_{r,\ 2}$, $F_{r,\ 3}$,	[1, 2, 2]

2) Model-Agnostic Hyperparameters: Regarding the model-agnostic hyperparameters, there exist multiple and are summarized in Appendix A alongside their values. However, one model-agnostic hyperparameter will be optimized, the number of kernels. This is done as the number of kernels is known to have an effect on the generalization ability of the model [59]. A higher number of feature maps produced caused by a higher number of kernels could allow for greater generalization ability of the model as it would allow for the understanding of a wider range of features, enhancing the model's ability to understand complex patterns and variations in the data though at the cost of computational power and efficiency. However, if the number of feature maps produced is too high, it could lead to overfitting as it may learn to understand noise and irrelevant details. Therefore, it is important to optimize this hyperparameter and the values to be tested will be 128 and 256 as it was the two different values used by models that were designed for MTS classification including ResNet [22], [23], [43]–[47].

A summary of the hyperparameters to be optimized along with the candidate values to be considered can be seen in Table III.

Table III: Hyperparameters to be optimized

Hyperparameter	Candidate Values
Number of Kernels	[128, 256]
Window Size, %	[20, 40, 60, 80, 100]

IV. Approach

Before delving into the approach of this study, it is first important to briefly summarise the different data splits that will be used to assist in the quantitative measure of the performance of the NN model. To be specific, there are three datasets into which the original dataset can be split:

- Training Dataset: The dataset split employed to train the model.
- Validation Dataset: The dataset split employed for hyperparameter tuning and to validate the model when it is training, i.e., to mitigate overfitting [60] and ensure the model's generalization capabilities.
- Testing Dataset: The dataset split employed to test the model on unseen samples.

A. Hyperparameter Optimization

The first step in the approach was to determine the most optimal value for the hyperparameters shown in Table III. This was achieved through the use of a subject-wise stratified k-fold Cross-Validation (CV) [49] where the given data set is split into k number of folds where each fold is used as a validation set at least once. Note that this will be done only with the use of the RW Data as multi-axis data is the main focus of this study. For this study, it was decided to set k as 5 as it allows for a balance in allowing for generalization while maintaining lower computational training times.

To conduct the hyperparameter optimization, the number of kernels was the first hyperparameter to be tuned. To do so, the *Window Size* was fixed to 80%. This value was chosen as an initial value due to the fact that in [46], the value of 80% yielded the highest accuracies for most datasets. After determining the validation accuracy for each of the candidate values for the number of kernels, the value that yielded the highest accuracy is chosen as the optimal value. Once the number of kernels was optimized, the *Window Size* hyperparameter was then tuned with the number of kernels fixed to the optimal value determined. Similarly, the validation accuracy for each of the candidate values for *Window Size* was determined with the value yielding the highest accuracy chosen as the optimal value. Finally, with the optimal value of each hyperparameter obtained, the number of trainable parameters was then set.

B. Performance Analysis

A visual plan of the performance analysis section of the approach is presented in Fig. 4. Phases 1 and 2 aim to test the model's performance within a single dataset, i.e. intra-dataset, whereas Phases 3 and 4 aim to test the model's performance amongst multiple datasets, i.e. inter-dataset.

1) Phase 1: The goal of Phase 1 is to determine the model's overall ability to classify skill levels of time traces obtained from dual-axis tracking tasks and compare it to the model used by De Jong in their research, ResNet [21], [22]. In order to allow a fair comparison of the two, the same hyperparameter values used for RN-XCM will also be applied to ResNet. In addition, the amount of training time required by the two models will be recorded to determine the extent to which RN-XCM is more compact than ResNet. To evaluate the overall performance of the models, two scenarios are to be established to determine the model's performance in each one: the "ideal" scenario and the "realistic" scenario. These scenarios are visualized in Fig. 5.

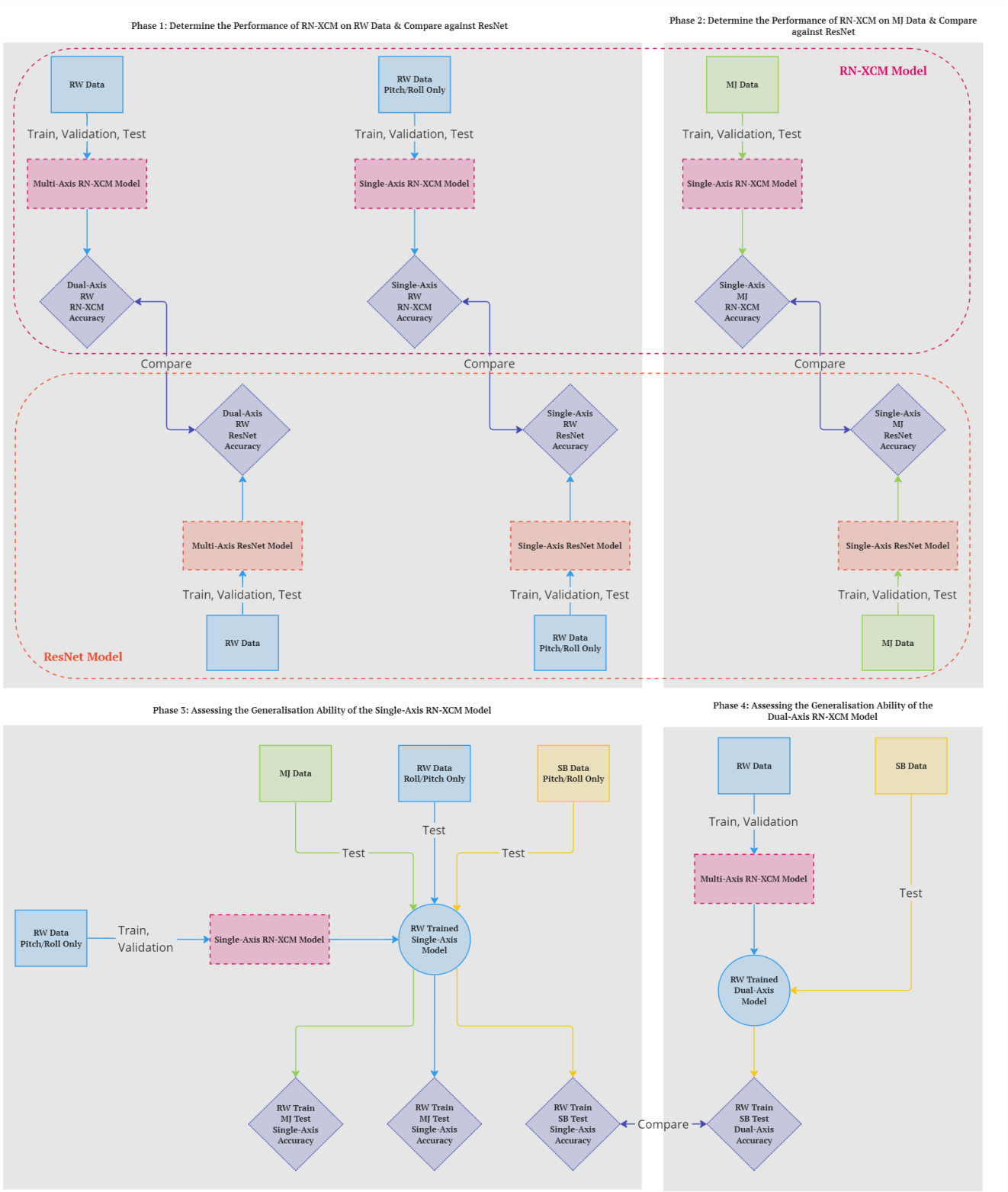

Figure 4: Phases of the performance analysis of the RN-XCM model.

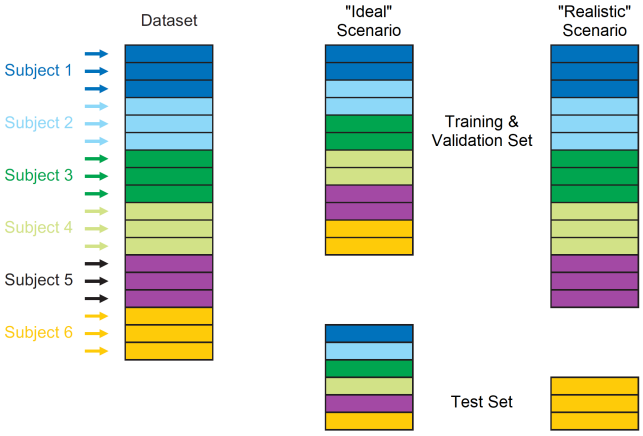

Figure 5: A sample dataset of 6 subjects being transformed to create the "ideal" and "realistic" scenarios, adapted from [49].

The "ideal" scenario aims to simulate the scenario in which the model is classifying the tracking run of an HO on whose data it has been trained. To create the "ideal" scenario, the processed dataset from the RW Data [26] will be shuffled and split into training data, validation data, and test data. The split will be performed with a ratio of 80/10/10 with 80% of the data being training data, 10% being validation data, and the final 10% being test data. In addition, in each split of the data, a time trace window of each HO will be present. This is to ensure that the model has indeed been trained with the run of an HO that it will be tested on. The testing accuracy produced will be used as the final "ideal" scenario accuracy.

The "realistic" scenario aims to simulate the most realistic scenario in which the model has to classify the tracking run of a HO with which it was never trained on. This is the most realistic scenario for the fact that if this model were to be deployed in the real world, it would be impossible for it to be trained on every single person that it would be used by due to the sheer number of people that would use it. As the model would be classifying a HO it was never trained on, **Hypothesis 1A (H.1A)** hypothesizes that the model will not perform as well as it would in the "ideal" scenario described prior. To create the "realistic" scenario, all the runs of a singular HO will be separated from the processed dataset in the RW Data [26] and will be used as the test dataset. The remaining dataset will then be shuffled and split into training data and validation data with a ratio of 80/20 where 80% of the data will be for training and the remaining 20% will be for validation. It will also be ensured that the training and validation data will contain at least a singular time trace window of the remaining HOs. This will be done for every single HO within the dataset and the average testing accuracy across all HO will be used as the final "realistic" scenario accuracy.

As preliminary analysis [25] found that RN-XCM performed similarly to ResNet when classifying a small sample of the RW Data in a "realistic" scenario while requiring 96.36% less training time, **Hypothesis 1B (H.1B)** hypothesizes that RN-XCM will also obtain similar classification accuracies to ResNet in both "ideal" and "realistic" scenarios while maintaining shorter training times.

The obtaining of "ideal" scenario accuracy and "realistic" scenario accuracy will be done for not only dual-axis data, but also single-axis data obtained by separating the RW Data [26] into a singular axis of either pitch or roll, i.e. RW Data Pitch Only and RW Data Roll Only. This was done in order to determine to what extent are the models able to classify the tracking data given only one axis of the multi-axis data. **Hypothesis 1C (H.1C)** hypothesizes that due to the presence of *Crossfeed*, i.e. the inability for the HOs to decouple the tasks per axis [24], feeding the model only axis will yield lower test accuracies as it does not present a complete image of the HO's multi-axial control behavior.

A visual plan of this phase of the approach is presented in the top left of Fig. 4.

2) Phase 2: The goal of Phase 2 is to determine RN-XCM's overall ability to classify skill level from time traces obtained from single-axis tracking tasks and compare it against the model used by De Jong in their research, ResNet [21], [22]. To assess the overall performance of the models, the accuracies of both the "ideal" and "realistic" scenarios will be obtained through a different dataset. Specifically, the dataset used in this phase will be the one used by De Jong [21], the MJ Data [8]. As stated before, this dataset is different from the RW Data used in Phase 1 as those data represent the HO control behavior in a single-axis tracking task rather than a dual-axis tracking task. As the model will be classifying HOs it was never trained on in the "realistic" scenario, **H.1A** is extended here where it is hypothesized that the model will not perform as well in the "realistic" scenario as in the "ideal" scenario. Similarly, as preliminary analysis [25] found that RN-XCM performed similarly to ResNet when classifying a small sample of the RW Data in a "realistic" scenario while requiring less training times, **H.1B** is extended here. A visual plan of this phase of the approach is presented in the top right of Fig. 4.

3) Phase 3: The goal of Phase 3 is to determine the model's generalization ability in single-axis tracking tasks given only one dataset to train on. To be specific, the model will be fully trained and validated with the entirety of the single-axis separated RW Data [26], i.e. RW Data Pitch Only and RW Data Roll Only, and test its ability to classify a dataset it has never seen and with different control task variables. The dataset that will be used as the test dataset will be the MJ Data [8], the opposite axis of the RW Data, and the SB Data [24]. They will be used to test the fully trained model separately and thus, there are three separate hypotheses to be tested in this phase. A visual plan of this phase of the approach is presented in the bottom left of Fig. 4.

Hypothesis 3A (H.3A) hypothesizes that if the trained model were tested against the MJ Data, it would produce more false positives than other types of classifications, that is the classifier will often classify "unskilled" labeled data as "skilled". This hypothesis stems from the assumption that the

NN model, after being trained with data obtained from dual-axis tracking tasks, will capture the cybernetic phenomena [24] caused by the presence of multiple axes to control, specifically overall performance degradation in dual-axis tracking tasks. **H.3A** further hypothesizes that if the model is trained on the MJ Data and tested on both of the singular axis data of the RW Data, more false negatives will be produced for the same reason, i.e., classifying "skilled" labeled data as "unskilled".

Hypothesis 3B (H.3B) hypothesizes that if the trained model on the pitch axis of the RW Data is tested on the roll axis of the RW Data, the classifier will classify the "skilled" runs in the test dataset as "unskilled". **H.3B** further hypothesizes that if the trained model on the roll axis of the RW Data is tested on the pitch axis of the RW Data, the classifier will classify the "unskilled" runs in the test dataset as "skilled". This is due to the phenomenon of *Axis Asymmetry* in which HO has better overall performance in one axis over the other, specifically the pitch axis over the roll axis [24].

Hypothesis 3C (H.3C) that if the trained model would be tested against the SB Data, it would correctly classify it as "skilled" as the SB Data contains tracking data of HOs after their training runs, i.e. their evaluation runs, and on a tracking task that is relatively "easier" than that of the RW Data due to the simpler control dynamics [24].

4) Phase 4: The goal of Phase 4 is to determine the model's generalization ability in dual-axis tracking tasks given only one dataset to train on. To be specific, the model will be fully trained and validated with the entirety of the RW Data [26] and test its ability to classify a dataset it has never seen and with different control task variables. The dataset that will be used as the test dataset is the SB Data [24]. Due to the similarity to the third scenario of Phase 3, **H.3C** will be extended here where it is hypothesized that the trained model will correctly classify the SB Data as "skilled". A visual plan of this phase of the approach is presented in the bottom right of Fig. 4.

V. Results

A. Hyperparameter Optimization

The result of the number of kernels optimization with *Window Size* set to 80% is shown in Table IV. Subsequently, the result of the *Window Size* optimization with the number of kernels set to 128, as it produced the highest accuracies, is shown in Table V.

Overall, the final optimized hyperparameter values found to yield the best performance out of the RN-XCM model are 128 for the number of kernels and 20% for the *Window Size* hyperparameter. Note that the optimal value for the number of kernels will be used for ResNet as well to allow for a fair comparison between the models. Regarding the validation accuracy of 81.20% obtained through these optimal hyperparameter values, it is a good indication of what can be expected regarding the test accuracies of the RN-XCM model in the "realistic" scenario with the RW Data. It is important to note though that the obtained "realistic" test accuracies will more than likely be lower than this value simply due to the fact that the model is repeatedly shown the validation dataset every epoch with the goal of improving itself in order to maximize the validation accuracies with each epoch. In contrast, for test accuracy, the model is tested against the test dataset only once. In addition, these validation accuracies are significantly lower than those obtained by De Jong of 87.95% when hyperparameter tuning the ResNet model on their no-motion dataset [21]. This however is due to the difference in the hyperparameter tuning method as they used an 80/20 random split of their dataset, which would mean that the ResNet model has seen at least one sample from each HO before seeing the validation dataset. In contrast, the stratified k-fold CV method utilized in this study meant that the model never saw any sample from HOs within the validation dataset, hence the lower validation accuracies.

With the optimal hyperparameters obtained, the final amount of trainable parameters of the two models was obtained with RN-XCM having 195,000 trainable parameters and ResNet having 505,000 trainable parameters. It can be seen that given the same hyperparameter values, RN-XCM is a much more compact NN model having 61.38% fewer trainable parameters.

Table IV: Validation accuracies of different numbers of kernels

Number of Kernels	Average Validation Accuracy, %
128	80.33
256	80.26

B. Performance Analysis

1) Phase 1: In Phase 1, the performance of the RN-XCM model on dual-axis tracking data, RW Data, is quantified, see Fig. 4. Between the two scenarios, it can be seen that for the complete dual-axis RW Data, the two NN models had an average drop in performance from the "ideal" to the "realistic" scenario of 14.83%, proving **H1.A**. On average, it can be seen that RN-XCM obtained extremely similar accuracies compared to ResNet. Specifically, for the "ideal" scenario, ResNet, on average, obtains accuracies 2.97% better than those obtained by RN-XCM. In contrast, in the "realistic" scenario, RN-XCM, on average, obtains accuracies 0.96% better than those obtained by ResNet. However, due to the fact that RN-XCM has 61.38% fewer trainable parameters, it requires 50% less training time than that required by ResNet. The RN-XCM model's ability to require less computational workload whilst

Table V: Validation accuracies of different values for *Window Size*

Window Size, %	Average Validation Accuracy, %
20	81.20
40	80.37
60	80.51
80	80.33
100	80.53

maintaining comparable performance to ResNet showcases the former's more efficient analyzing approach and proves **H.1B**. Lastly, it can be seen that the accuracies obtained for the RW Data Pitch Only and Roll Only are consistently worse than those obtained for the complete RW Data. Specifically, there was an average drop of 3.08% for the "ideal" scenario and 1.39% for the "realistic" scenario, proving **H.1C**.

Table VI: Phase 1 accuracies obtained by RN-XCM and ResNet

Scenario	Dataset	RN-XCM	ResNet
"Ideal" Scenario, %	RW Data	93.50	95.45
	RW Data Pitch Only	89.97	93.44
	RW Data Roll Only	89.33	92.83
"Realistic" Scenario, %	RW Data	80.16	79.14
	RW Data Pitch Only	78.35	77.73
	RW Data Roll Only	79.09	77.86

Table VII: Phase 1 training times of RN-XCM and ResNet for the "ideal" and "realistic" scenarios

Scenario	RN-XCM	ResNet
"Ideal" Scenario, s	1,720	3,440
"Realistic" Scenario, s	70,711	141,422

To better understand the obtained results, the confusion matrices for the "ideal" and "realistic" scenarios for the RN-XCM model of the complete dual-axis RW Data are shown in Table VIII and Table IX respectively. When looking at the trend of the majority of the confusion matrices of both models for both the "ideal" and "realistic" scenarios, it can be seen that although there indeed was a difference in the majority of incorrect classifications, i.e. the labeling "unskilled" runs as "skilled" and in the latter scenario and vice versa, the difference is minuscule at an average of 1.08%. This showcases that there is likely no difference in the classification strategies employed by the RN-XCM model when classifying data in the "ideal" and "realistic" scenarios.

Table VIII: Confusion matrix of the RN-XCM model in the "ideal" scenario with the complete DA RW Data

		Predicted	
		U	S
Actual	U	47,170 46.11%	3,811 3.73%
	S	2,836 2.77%	48,479 47.39%

To add to the obtained results, the "realistic" scenario was extended such that each out-of-sample test was done not only on the labeled runs, i.e., the first and last 20 runs but throughout the entire unlabeled 100 runs. Although this will not yield any classification accuracies due to the lack of labels, the purpose of this test was to determine whether the model would increase the number of "skilled" predictions it would give with an increase in run number. This is shown in Fig. 6

Table IX: Confusion matrix of the RN-XCM model in the "realistic" scenario with the complete DA RW Data

		Predicted	
		U	S
Actual	U	416,146 40.68%	95,334 9.32%
	S	107,599 10.52%	403,881 39.48%

where the average model output of every subject per run is plotted over every singular tracking run number. From this figure, it can be seen that with an increasing number of runs, the model shows an overall increase in "skilled" classification. When plotting Fig. 6, vertical lines were drawn at run numbers 25, 50, and 75 to coincide with the final runs of each training day of the RW Data [26]. As can be seen, the number of "skilled" classifications tends to dip as the runs get closer to the end of each training day. In addition, it was also seen that the first run of each training day was much worse than the last run of the previous training day. When looking at the average model output per training day, as shown in the four large values on top between each vertical line, it can be seen that it increases with every passing training day. To be specific, the model detects an average increase in skill level by 18.33% with every training day. Overall, this showcases that the RN-XCM model is able to effectively recognize an improvement of skill of the HOs with an increase in runs alongside some form of HO fatigue due to the detected drop in skill level at the end of the training days and loss of skill by the HO due to the detected drop in skill level at the beginning of each training day.

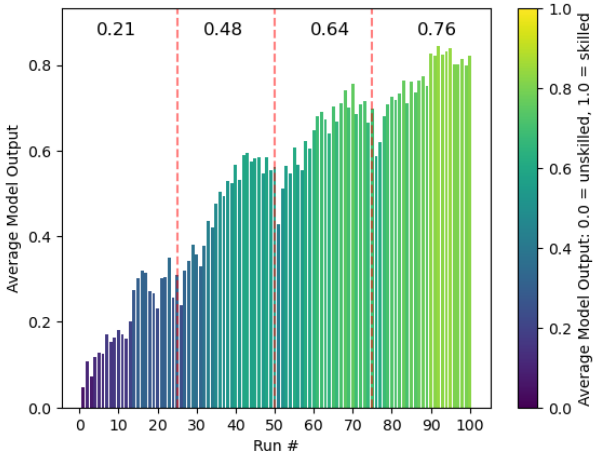

Figure 6: Average model output for the complete dual-axis RW Data with an increasing number of runs with vertical lines at runs 25, 50, and 75. The average output per training day, i.e. for every 25 runs, are shown on the top of each respective day.

2) Phase 2: In Phase 2, the performance of the RN-XCM model on the single-axis MJ Data is quantified. This is done with the same steps as was done in Phase 1. Table X shows the accuracies of both the RN-XCM and ResNet model [22] in both the "ideal" and "realistic" scenarios. Similar to what was observed in Phase 1, there was a drop in the accuracies obtained from the "ideal" scenario to those obtained from the "realistic" scenario, further proving **H.1A**. However, it can be noticed that the drop in Phase 2 is much higher than those found in Phase 1. Specifically, Phase 2 had an average drop of 25.59% for both models compared to the average drop of 14.83% for both models in Phase 1. This difference in the drop of accuracies implies that both NN models struggled to accurately classify the skill level of never-before-seen HO tracking data originating from the MJ Data compared to those originating from the RW Data.

As was found in Phase 1, in the "ideal" scenario, ResNet obtained an accuracy 2.80% higher than that obtained by RN-XCM, and in the "realistic" scenario, RN-XCM obtained an accuracy 0.38% higher than that obtained by ResNet. Similarly, looking at Table XI, the RN-XCM required 51.72% less training time than that required by ResNet. This reduced training time required whilst maintaining similar accuracies again showcases RN-XCM's more efficient analyzing approach, proving **H.1B**.

Table X: Phase 2 accuracies obtained by RN-XCM and ResNet

Scenario	Dataset	RN-XCM	ResNet
"Ideal" Scenario, %	MJ Data	91.34	94.14
"Realistic" Scenario, %	MJ Data	63.64	63.25

Table XI: Phase 2 training times of RN-XCM and ResNet for the "ideal" and "realistic" Scenarios

Scenario	RN-XCM	ResNet
"Ideal" Scenario, s	560	1,160
"Realistic" Scenario, s	7,270	15,060

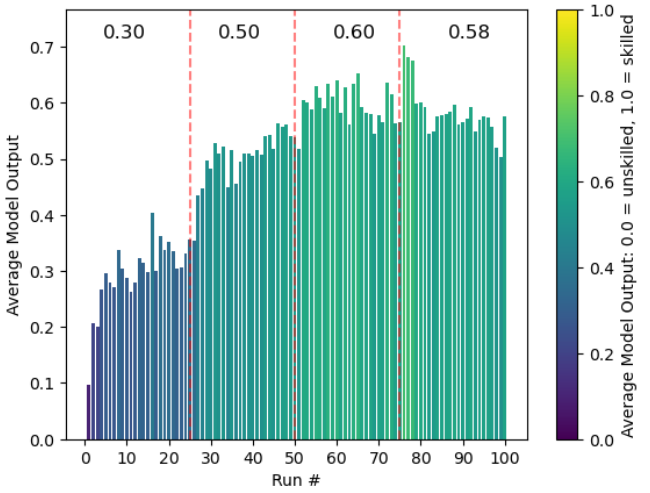

Figure 7: Average model output for the complete single-axis MJ Data with an increasing number of runs with vertical lines at runs 25, 50, and 75. The average output per training day, i.e. for every 25 runs, are shown on the top of each respective day.

The "realistic" scenario for the MJ Data was similarly extended such that each out-of-sample test was done through the entire unlabelled 100 runs. Fig. 7 showcases the average model output of every subject per run over every singular run. Similar to Fig. 6, vertical lines were added at runs 25, 50, and 75, when plotting Fig. 7 to coincide with the final runs of each training day of the MJ Data [8]. However, unlike the result obtained in Fig. 6, the average model output does not consistently showcase an overall increase in "skilled" classifications with an increasing number of runs as in the last 50% of runs, it can be seen that from the average model output of each training day, shown through the large numbers between each vertical line, the model detects that skill level stagnates at around 59%. In addition, there is no consistent dip in average model output as the runs get closer to the end of the training days, the red vertical lines, with detected loss of skill level at the beginning of the training days as was found in Fig. 6. This further implies that both NN models struggled to accurately classify the skill level of never-before-seen HO tracking data originating from the MJ Data compared to those originating from the RW Data.

3) Phase 3: In Phase 3, RN-XCM's generalization ability as a single-axis classifier is quantified. To do so, the performance of the model and the nature of its predictions when trained and tested on data of different natures. The first of these results is the scenario of the model being trained on either the Pitch-Only or Roll-Only RW Data and then tested on the MJ Data. The confusion matrices of the results obtained are presented in Table XII and Table XIII, respectively. As hypothesized by **H.3A**, a majority of "unskilled" samples from the MJ Data were classified as "skilled" at 96.68% and 99.56% of "unskilled" runs were labeled as "skilled" in Table XII and Table XIII, respectively. In addition, it can also be seen that respectively, 98.24% and 99.60% of "skilled" runs were labeled correctly as "skilled".

When the opposite situation was tested, training the model on the MJ Data and then testing on either RW Data Pitch Only or RW Data Roll Only, the opposite effect was found. A majority of the "skilled" samples were classified as "unskilled", as seen in Table XIV and Table XV at 70.12% and 72.94% each respectively. In addition, it can also be seen that respectively, 82.74% and 80.20% of "unskilled" runs were correctly labeled as "unskilled". This further proves **H.3A** in that due to the phenomenon of *Performance Degradation*, "skilled" runs in dual-axis tracking tasks correspond to comparatively more "unskilled" than "skilled" runs in single-axis tracking runs. However, it is important to note that unexpectedly, the amount of mislabeled data is not as high as those obtained in the previous situation shown in Table XII and Table XIII and therefore, some form of asymmetry regarding the obtained

results exist.

Table XII: Confusion matrix of the RN-XCM model trained on RW Data pitch only and tested on MJ Data

		Predicted	
		U	**S**
Actual	**U**	5,369 1.66%	156,151 48.34%
	S	2,854 0.88%	158,666 49.12%

Table XIII: Confusion matrix of the RN-XCM model trained on RW Data roll only and tested on MJ Data

		Predicted	
		U	**S**
Actual	**U**	699 0.22%	160,821 49.78%
	S	670 0.20%	160,850 49.80%

Table XIV: Confusion matrix of the RN-XCM model trained on MJ Data and tested on RW Data pitch only

		Predicted	
		U	**S**
Actual	**U**	423,161 41.37%	88,319 8.63%
	S	358,700 35.06%	152,780 14.94%

Table XV: Confusion matrix of the RN-XCM model trained on MJ Data and tested on RW Data roll only

		Predicted	
		U	**S**
Actual	**U**	410,192 40.10%	101,288 9.90%
	S	373,081 36.47%	138,399 13.53%

The second scenario was for the model to be trained on the Pitch-Only RW Data and then tested on the Roll-Only RW Data and vice versa. The confusion matrix of these results is presented in Table XVI and Table XVII respectively. As hypothesized by **H.3B**, the majority of the "skilled" runs in the roll axis were classified as "unskilled" when the model was trained with data from the pitch axis at 72.08%. Similarly, the majority of the "unskilled" runs in the pitch axis were classified as "skilled" when the model was trained with the data from the roll axis at 67.06%. This supports the hypothesis **H.3B** in that due to the phenomenon of *Axis Asymmetry* "skilled" runs in the roll axis will be relatively more "unskilled" than "skilled" runs in the pitch axis.

The third and final scenario was for the model to be trained on either the Pitch-Only or Roll-Only RW Data and then tested on the Pitch-Only or Roll-Only SB Data and the opposite of Roll-Only or Pitch-Only SB Data. As the SB Data had only one class, "skilled", no confusion matrix was obtained. It was found that in all scenarios, the model correctly classified with 100% accuracy that all 323,040 samples in the SB Data were "skilled", as hypothesized by **H.3C**.

Table XVI: Confusion matrix of the RN-XCM model trained on RW Data pitch Only and tested on RW Data roll Only

		Predicted	
		U	**S**
Actual	**U**	479,440 46.87%	32,040 3.13%
	S	368,656 36.04%	142,824 13.96%

Table XVII: Confusion matrix of the RN-XCM model trained on RW Data pitch only and tested on RW Data roll only

		Predicted	
		U	**S**
Actual	**U**	168,470 16.47%	343,010 33.53%
	S	3,502 0.34%	507,978 49.66%

4) Phase 4: In Phase 4, the last phase, RN-XCM's generalization ability as a dual-axis classifier of different task variables is quantified. Similar to Phase 3, this is done by obtaining the performance of the model and the nature of its predictions when trained and tested on data from two different experiments. As there are only two datasets with dual-axis tracking data, only one scenario exists, training the model with the RW Data and then testing it on the SB Data. As stated before, the SB Data had only one class and therefore no confusion matrix was obtained. Similar to what was obtained in the third scenario of Phase 3, the model would correctly classify with 100% accuracy and confidence that all 323,040 samples in the SB Data were "skilled", further proving **H.3C**.

VI. Discussion

This paper aimed to implement an explainable deep-learning approach in order to effectively classify the skill level of HOs in dual-axis tracking tasks based purely on their control behavior. The result of this work resulted in the inception of a new multivariate NN model built specifically for the aforementioned purpose, RN-XCM. When given 1.2 s windows of tracking data in dual-axis tracking tasks, the classifier is able to obtain a test accuracy of 93.50% when shown tracking data of participants it was already trained on and 80.16% when shown tracking data of participants it has never been trained on. This was achieved while having less than half the parameters of the next best NN model, ResNet [22], as the former contains 195,000 trainable parameters and the latter

contains 500,000 trainable parameters. This results in the RN-XCM requiring around 50% less training time compared to ResNet. In the paragraphs to follow, the results presented in Section V will be reviewed and reflected upon to discuss their relevance, shortcomings, and potential improvements.

A. Hyperparameter Optimization

For the hyperparameter optimization, it was found that 128 kernels and an XCM *Window Size* of 20% resulted in the most optimal classification performance. This optimization was done by first fixing the window size to 80% and determining which number of kernels was best. With the optimal number of kernels obtained, the window size was optimized with the number of kernels fixed to the optimal one. This process of optimizing hyperparameters, however, could have been done more extensively by performing a grid search, i.e., testing on every combination of the number of kernels and *Window Size* values. In addition, the hyperparameter optimization step could have been done for more hyperparameters as was done by De Jong [21]. As this would result in an incredibly large search space of hyperparameter values, 480 unique combinations of parameter values to be exact, a grid search would result in a significant amount of computational time. Although the expected gains from such hyperparameter tuning can be seen as very small, as De Jong only reported a performance increase of 1.60% [21], if extracting the utmost performance out of the NN model is the goal, a more efficient hyperparameter tuning can be performed by utilizing a random search [61].

B. Performance Analysis

1) Phases 1 & 2 (Intra-Dataset): For Phase 1 of the performance analysis of the RN-XCM model, it was found that when utilizing the complete RW Data, RN-XCM obtained an "ideal" accuracy of 93.50% and a "realistic" accuracy of 80.16%. Subsequently, for Phase 2, it can be seen that with the MJ Data, single-axis tracking data, RN-XCM obtained an "ideal" accuracy of 91.34% and a "realistic" accuracy of 63.64%. It can be seen that as expected by **H.1A**, the "ideal" scenario accuracies are consistently higher than those obtained for the "realistic" scenarios. However, the drop in accuracies between the scenarios show greater disparity between the datasets used. To be specific, when using the complete RW Data on the RN-XCM model, i.e. Phase 1, the drop is 13.34% whereas when using the MJ Data on the RN-XCM model, i.e. Phase 2, the drop is 27.70%. This suggests that De Jong's belief that the labeling method of the runs was the cause of this performance was not the case [21]. Instead, it suggests that one of two reasons could be the cause of the difference in the drop in performance (or a combination of both).

First, the RW Data consisted of control behavioral data of a dual-axis tracking task whereas the MJ Data consisted of control behavioral data of a single-axis tracking task. This is significant as dual-axis tracking tasks cause lower crossover frequencies compared to single-axis tracking tasks given the exact same dynamics [24]. Adding to this, the controlled dynamics of the task for the RW Data [26] are significantly more difficult than that found in the MJ Data [8] which would only accentuate the drop in crossover frequencies [24]. Therefore, HOs are unable to respond as quickly to changes in the tracking task, causing greater noise in the control behavior of the input signals in the RW Data compared to those found in the MJ Data [24]. This greater noise within the input signals would therefore allow for easier classification of skill level compared to input signals with much less variability than those obtained in single-axis tracking tasks that is the MJ Data. This highlights that while single-axis tasks offer ease of experimentation and analysis [9], multi-axis tasks present a more robust approach for skill-level classification using NN due to the heightened variability in input and error signals.

Second, the RW Data consisted of control behavioral data from 38 participants whereas the MJ Data consisted of control behavioral data from 12 participants. The smaller sample size in the MJ Data would cause the model to have less understanding of the variability of control behavior amongst HOs and the general underlying patterns that dictate whether a run was "skilled" or "unskilled". Therefore, as the RW Data had 38 participants, the models had a better understanding of the aforementioned variability and therefore, recognized the general underlying patterns that dictate whether a run was "skilled" or "unskilled". A future study to determine to what extent this performance drop decreases with an increase in participants to train would be important to determine whether there is a threshold to how low the difference can be or if it is entirely possible to completely negate this drop. In contrast, the RW Data could also be limited to only 12 subjects, to equal the number of subjects in the MJ Data, and determine whether the model's performance with said limited RW Data mimic those of the complete RW Data in Phase 1 or the MJ Data in Phase 2.

For both Phases 1 and 2, ResNet consistently obtained better "ideal" accuracies compared to RN-XCM, however, performed worse in "realistic" accuracies. Though it could be argued that ResNet's 2.97% and 2.80% better performance in the "ideal" scenario for Phases 1 and 2 respectively while only performing 0.96% and 0.39% worse in the "realistic" scenario compared to the RN-XCM model deems the former as the better model, this is not entirely true. This is for the fact that ResNet achieves this while requiring 2.59 times the trainable parameters and on average, 2.04 times the training time required by RN-XCM. This showcases the more efficient analyzing method of the RN-XCM model which allows it to be a more compact model while achieving similar accuracies to ResNet as was hypothesized in **H.1B**. In addition, ResNet's worse performance in the "realistic" scenario can be attributed to the difference in the depth of the models. To be specific, although the depth of ResNet allows for better performance [53] as it allows for the understanding of complex patterns in the data [22], it also allows for the model to understand noise or memorize certain behavioral aspects of each HO. This, therefore, causes ResNet's better performance in the "ideal" scenario and worse performance in the "realistic" scenario.

When tested on the RW Data Pitch Only and RW Data

Roll only, the same trend occurs with ResNet obtaining better accuracies in the "ideal" scenario and RN-XCM obtaining better accuracies in the "realistic" scenario. However, it can be seen that accuracies for both datasets in both scenarios are consistently lower than those obtained when using the complete RW Data, with an average drop of 3.08% in the "ideal" accuracy and 1.39% in the "realistic" accuracy as was hypothesized by **H.1C**. This shows that in order for any NN model to create accurate classifications, it requires signals in both axes rather than one over the other, i.e., a multi-axis classifier should always be used on multi-axis data.

Lastly, for Phase 1, it was seen that the RN-XCM model was able to correctly classify an increasing amount of skilled runs with an increase in training runs. Alongside this, the model was able to detect higher overall skill levels with every passing training day. The model was also able to detect the subtle effects of fatigue on the HOs as they got closer to the final runs of their respective training days due to the small dip in average model output as the runs got closer to the end of the training days. Finally, the model detected a loss in skill level between each training day as the detected skill level for the first run in each training day is consistently worse than the last run of the previous training day, showcasing the model's ability to detect a form of subconscious "forgetting" of skill level by the HOs. From these observations, it is evident that the RN-XCM skill level classification abilities are able to recognize the complex nature of skill development and the impact fatigue has on HOs. In addition, these findings put into perspective the importance of the training regimes employed for HOs in tracking tasks and highlight the potential for optimization of these regimes to enhance skill acquisition outcomes.

In contrast, in Phase 2, the RN-XCN model was not able to classify an increasing amount of skilled runs with an increase in training runs as well as was done with the RW Data in Phase 1. To be specific, although it correctly detected an increase in skills with runs for the first 2 training days, i.e. the first 50 runs, the amount of detected "skilled" runs stagnated and eventually decreased in the amount of detected skilled runs in the last 2 training days, i.e. the last 50 runs. The cause of this could be due to two factors (or a combination of both). The first possible cause could be due to less noise in control data and the lower number of participants within the MJ Data leading to incorrect classifications. The second possible cause could be the fact that after an increase in skill in the first 2 days, in the last 2 training days, the HOs stagnated in performance and eventually became fatigued due to the extensive duration of the training phase, leading to the decrease in performance in the last day. This stagnation and decrease in the number of skilled runs were also faced by De Jong when using ResNet [21] as their values when plotting the average model output of every subject per run over a range of run indices is extremely similar to those using RN-XCM, as is shown in Appendix B. Overall, a few key implications can be stated. Firstly, it brings back the previous implication that multi-axis tasks present a more robust approach for skill level classification using NN due to the heightened variability in input and error signals. Second, it highlights the importance of having a large number of test subjects to allow little to no room for doubt that the results obtained are not due to variability within test subjects, especially in studies, such as the current one, that involve the extremely variable nature of human behavior. Lastly, it also brings to light the important training regimes employed for HOs in tracking tasks and how they should be optimized such that HO skill level should always increase. Rarely should the training regimes cause a stagnation of skill level as that would imply a misuse of resources and never should it cause a decrease in skill level.

2) Phases 3 & 4 (Inter-Dataset): In Phase 3 of the performance analysis of the RN-XCM model, it was found that in the scenario of training the model on either the Pitch-Only or Roll-Only RW Data and then testing it on the MJ Data, the model would consistently classify "unskilled" runs from the MJ Data as "skilled", which was as hypothesized by **H.3A**. As expected, when the opposite was done, training the model on either the MJ Data and then testing it on the Pitch-Only or Roll-Only RW Data, the model would then classify the "skilled" runs in the RW Data as "unskilled". Overall, this was due to the phenomenon of *Performance Degradation* in dual-axis tracking tasks compared to single-axis tasks due to lower crossover frequencies, and an increase in error variance [24].

It was also observed that although hypothesis **H.3A** was indeed correct, there exists some asymmetry regarding the hypothesized results. To be specific, when training with RW Data and testing with MJ Data, an average of 98.12% of "unskilled" labeled data were mislabeled as "skilled". In contrast, when training with MJ Data and testing with RW Data, an average of 71.53% of "skilled" labeled data were mislabeled as "unskilled", a stark difference from the reverse scenario. This asymmetry could be caused by a combination of two factors.

The first possible factor is the fact that the difference in samples in the MJ Data compared to the RW Data. To be specific, the MJ Data contains 68.42% fewer samples than the RW Data. Therefore, in the scenario of having the model train on the MJ data, the model had relatively much fewer samples to train on. This causes the model to be unable to learn the underlying features and patterns necessary for accurate generalization as effectively as it would when using the much larger RW Data as a training dataset.

The second possible factor is linked to the aforementioned greater variability within the error and control signals within the RW Data compared to the MJ Data. When training the NN model with the more variable RW Data and then testing on the less variable MJ Data, the model was trained such that it extract underlying features and patterns that are common across the many variations of the data, allowing it to generalize better. Thus when tested on less variable data, the model was able to obtain better results. In contrast, in the opposite scenario, the model is trained on the less variable MJ Data, therefore the model is unable to learn all the underlying features and patterns necessary for accurate generalization.

Therefore, after training on the MJ data and then testing on the more variable RW Data, the model obtains worse results. Taking into account the previous fact that in the latter scenario, the MJ Data is too small to be effectively used as a training dataset, it can be seen that the asymmetrical results were to be expected.

In the scenario of training the model on the Pitch-Only RW Data and then testing it on the Roll-Only RW Data and vice versa, it was found that in the first case, the model would classify the majority of the "skilled" runs as "unskilled" and for the second case, the majority of the "unskilled" runs as "skilled". This was as **H.3B** hypothesized as it was found that in dual-axis tracking tasks, there existed the phenomena *Axis Asymmetry*, which states that HOs tend to have higher crossover frequencies and a decrease in error variance in the pitch axis over the roll axis [24].

In the final scenario of training the model on the Pitch-Only or Roll-Only RW Data and then testing it on the Pitch-Only or Roll-Only SB Data and the opposite of Roll-Only or Pitch-Only SB Data, it was found that the model would consistently label all of the SB runs correctly as "skilled" with 100% accuracy. Though **H.3C** hypothesized that this would occur due to the fact that the runs from the SB Data were after the training runs and had relatively easier to-control system dynamics, the 100% accuracy brought a level of skepticism as it could have been caused by the fact that the model is simply guessing correctly by pure chance. However, the obtained results align with expectations due to several factors. Firstly, the SB Data comprised of runs after the training phase, which are relatively more "skilled" than any "skilled" runs within the training phase that were used to train the classifying NN model. Moreover, the control task of the dataset is simpler than those used in the RW Data that was used to train the NN model [24], [26]. Taking these factors into account, the achieved 100% accuracy on the SB Data provides a reliable indication of the model's proficiency, rather than mere chance, reinforcing the validity of the hypothesis.

Overall, for the generalization of single-axis tracking tasks, i.e. Phase 3, it can be seen that the RN-XCM model was unable to effectively generalize as a single-axis classifier due to the emergence of cybernetic phenomena stemming from the intricate multi-axis nature of the RW Data. To elaborate, scenario 1 was characterized by the phenomenon *Performance Degradation*, while scenario 2 exhibited *Axis Asymmetry*. In contrast, in the context of scenario 3, the model managed to achieve successful generalization. It is worth noting that in instances where the model was trained on the pitch axis of the RW Data and subsequently tested on the roll axis of the SB Data, and vice versa, the phenomena of *Axis Asymmetry* would be prevalent, and would therefore theoretically lead to misclassifications. However, no such misclassification occurred. This most likely can be attributed to the elevated skill level inherent in the SB Data, which falls within the upper range of what is deemed 'skilled' runs as these runs were conducted for assessment purposes rather than training [24]. Consequently, the presence of *Axis Asymmetry* was mitigated, allowing for ease of generalization. Therefore, any future research focused on developing a single-axis skill-level classifying NN model that would be able to generalize across diverse tracking task attributes should delve further into methods that enable the NN model to effectively consider these cybernetic phenomena. The ultimate goal would then be to reduce and mitigate the misclassification stemming from these phenomena.

In Phase 4 of the performance analysis of the RN-XCM model, it was found that in the scenario of training the model in the complete RW Data and then testing it on the complete SB Data, the model would, similar to the final scenario in Phase 3, correctly classify all the runs as "skilled" with 100% accuracy as was hypothesized by **H.3C**. As was explained in Phase 3, the origin of the data supports the obtained results. From these results, the model was able to generalize as a dual-axis classifier due to the fact the cybernetic phenomena that caused the misclassifications in Phase 3 are neither present nor will they ever affect the model as a multi-axis classifier. Therefore, the only factors that would have an effect on the generalization performance of the multi-axis classifier would be the difficulty of the task within the test dataset relative to the those of the training dataset. As is the case for Phase 4, the task responsible for the test dataset was relatively easier than that of the training dataset [24], [26]. In addition, the types of runs used for training and testing will also have an effect as in this case, the test runs, i.e. the SB Data runs, were in the upper range of what is considered to be "skilled" as they were evaluation runs and the training runs, i.e. the RW Data runs, were training runs. Therefore, future research focused on developing a dual-axis skill-level classifying NN model that would be able to generalize across diverse tracking task attributes would have to delve further into what extent the aforementioned two factors would have an effect on the generalization performance of the model.

C. Future Work

The results of this research present a compelling argument for the use of RN-XCM in order to classify the skill level of HOs in multi-axis tracking tasks. The achieved 93.50% accuracy in the "ideal" scenario and 80.16% accuracy in the "realistic" scenario whilst requiring only 50% the training of a similarly performing model, ResNet, presents itself as such. The high accuracy in the "ideal" scenario suggests that at its current state, it is ready to act as a skill-level classifier for already-seen pilots. However, the true test of generalization ability lies in how well the model can handle new, diverse, and unseen pilots while still maintaining good accuracy. The relatively high "realistic" accuracy showcases the model's competence as a one-size-fits-all classifier, though the drop in accuracy from the "ideal" scenario to the "realistic" scenario is a key point of interest for future works. Specifically, it would be interesting to see to what extent could this drop in performance for dual-axis tracking tasks be mitigated with an increase in the number of HOs in the available dataset as it was seen that an increase in HOs had a positive effect on the "realistic" scenario from the 12 subjects in the MJ Data to

the 38 subjects in the RW Data. Another method to explore in order to reduce this drop in performance is to extend the use of resolution blocks within the RN-XCM model to the path that extracts information relative to the input variables. This however will extend the training time with no certainty that the performance gained will justify the increase in training time.

When reflecting on the datasets used for this research, a few points of concern arose regarding the MJ Data and the SB Data. For the former, it would have been ideal if there were a larger number of participants to determine whether the performance dip in the "realistic" scenario was caused by the task itself or due to the significantly lower number of participants. For the latter, it would have been ideal if an extensive training phase was present and recorded to allow for a more extensive analysis using the said dataset as the lack of an extensive training phase resulted in an unsuccessful analysis when training the RN-XCM model with the SB Data and then testing on the RW Data. Therefore, any recreation of this research would greatly benefit from recreating the MJ and SB Data with the aforementioned improvements in mind.

The use of the three different datasets brought with it another point of interest in the form of the unique system dynamics used in each dataset. It is of interest to determine if the RN-XCM model performance would have been different if the system dynamics between the datasets were not as unique. Specifically, what would occur if Phases 3 and Phase 4 were repeated with data from a task in which the single-axis task that was performed for the MJ data had the same pitch dynamics as those used in RW Data and a task in which the dual-axis task that was performed for the SB Data has the same dynamics in both axis as the pitch dynamics in the RW Data. It is hypothesized that due to the extreme similarity in the task dynamics, the mislabeling of data found in Phase 3 and 4 due to the phenomenon of *Performance Degradation* would be stronger as the NN model would better recognize "unskilled" and "skilled" runs of the system dynamics it was trained with.

The current "realistic" scenario within this research was to determine the RN-XCM model's ability to generalize amongst different HOs as a one-size-fits-all classifier. It would be of significant interest to extend this to determine the RN-XCM model's ability to generalize amongst different tracking tasks. As seen in Phase 3, it was seen that the NN model was not able to generalize well due to the cybernetic phenomena that occur with multi-axis tracking data. Therefore, any future research that involves creating a skill-level classifying model that is able to generalize between different task variables has to be table into these aforementioned phenomena.

Lastly, as this study showcases the NN model's competence at classifying skill levels in dual-axis tracking tasks, it would be of considerable interest to determine the model's ability to classify the skill level of HOs in a full 6-degree-of-freedom task such as the controlling of a helicopter. Although this is an extremely complicated task, this increase in difficulty would theoretically allow for greater variation between "unskilled" and "skilled" control behavior which would allow for easier classification of skill level. Therefore, it is theorized that in doing so, the NN model would either obtain better accuracies or in the worst-case scenario, maintain similar accuracies as was obtained in this research for the RW Data.

Overall, the success of the use of RN-XCM in the skill classification of pilot behavior presents a compelling argument for the use of this NN model in a variety of use-case scenarios. Firstly, it would allow for real-time warning of skill degradation to pilots when controlling an aircraft. This would ensure that pilots are never worse in performance than what is expected of them. If incorporated alongside methods that would allow for an explainability aspect to the NN mode, such as Gradient-Based Class Activation Mapping (Grad-CAM) methods [62]–[64], the model would also allow for real-time feedback on pilot control behavior which would be substantial for not only real-time usage but also for training of inexperienced pilots which was explored as shown in Appendix C. The online usage of the NN skill classification model would also allow for adaptive automation. For example, through the use of haptic feedback, it could allow for adaptive automation in the form of stronger haptic feedback with lower detected skill levels and vice versa to allow for seamless human-machine interaction [65]. Lastly, as the NN model allows for high classification accuracy while requiring a low computational workload, it is recommended that for any future research that involves the use of NNs to classify human pilot behavior, RN-XCM should be utilized, especially for the use case on already seen HOs.

VII. Conclusion

This paper set out to incorporate DL in order to provide a model that would have the capabilities to classify the skill level of pilots in real-life scenarios in real-time. In doing so, a new compact CNN for MTS pilot behavior classification was produced in the form of RN-XCM. When applying the model to dual-axis tracking task data, a task that closely mimics the tasks faced by a pilot controlling an aircraft, the model was able to correctly classify the HO's skill level as either "unskilled" or "skilled" with a test accuracy of up to 93.50% if the control data came from a HO the model has already been trained on. When evaluating the model as a one-size-fits-all classifier, i.e., classifying tracking runs of HO the model has never seen, the model was able to maintain a relatively high accuracy of 80.16%, suggesting its competence as such. Though these accuracies are extremely similar to the next best model, ResNet, RN-XCM shines in its ability to obtain such accuracies with 61.38% less trainable parameters resulting in the model requiring 50% less training times compared to ResNet. The drop in accuracy from the "ideal" to "realistic" scenario, however, can be attributed to the fact HO control behavior varies greatly from one HO to another and therefore, to reduce this drop in performance, a greater pool of HOs to train on could allow for an improvement in the model's performance as a one-size-fits-all classifier.

When looking at the performance of both the RN-XCM and ResNet models on both multi-axis and single-axis data, it can be seen that both models perform much better when classifying the former than the latter. Specifically, when comparing the results of the models obtained from the dual-axis RW Data to the single-axis MJ Data, the models on average obtained 1.74% higher accuracies in the "ideal" scenario and 17.88% higher accuracies in the "realistic" scenario. This suggests that although single-axis tasks might be simpler to conduct experiments on and analyze, multi-axis tasks allow for better classification of skill levels for NNs due to greater variability in input and error signals for these tasks. When looking at the performance of the RN-XCM model when trained and tested on different datasets, it can be seen that the model was not able to effectively classify skill levels, obtaining accuracies less than 67%, due to the phenomena stemming from the existence of multiple controllable axes. When these phenomena are not present or when the skill level of the test dataset is in the upper range of "skilled", the model faced no difficulty in effectively classifying skill levels obtaining accuracies of 100%.

Overall, this new model allows for a deep learning approach in the online classification of the skill level of pilot control behaviors never before seen by other methods or NN models, aiding in the progression of increased safety within the aviation industry and human-machine interaction synergy.

REFERENCES

[1] L. Bainbridge, "Ironies of automation," *Automatica*, vol. 19, no. 6, pp. 775–779, 1983. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0005109883900468

[2] T. Sheridan and R. Parasuraman, "Human-automation interaction," *Reviews of Human Factors and Ergonomics*, vol. 1, pp. 89–129, 06 2005.

[3] A. Calvi, F. D'Amico, L. B. Ciampoli, and C. Ferrante, "Evaluation of driving performance after a transition from automated to manual control: a driving simulator study," *Transportation Research Procedia*, vol. 45, pp. 755–762, 2020, transport Infrastructure and systems in a changing world. Towards a more sustainable, reliable and smarter mobility.TIS Roma 2019 Conference Proceedings. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352146520301514

[4] J. C. de Winter, R. Happee, M. H. Martens, and N. A. Stanton, "Effects of adaptive cruise control and highly automated driving on workload and situation awareness: A review of the empirical evidence," *Transportation Research Part F: Traffic Psychology and Behaviour*, vol. 27, pp. 196–217, 2014, vehicle Automation and Driver Behaviour. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1369847814000904

[5] S. Casner, R. Geven, M. Recker, and J. Schooler, "The retention of manual flying skills in the automated cockpit," *Human factors*, vol. 56, 05 2014.

[6] D. Pool and P. Zaal, "A cybernetic approach to assess the training of manual control skills," *IFAC-PapersOnLine*, vol. 49, no. 19, pp. 343–348, 2016, 13th IFAC Symposium on Analysis, Design, and Evaluation ofHuman-Machine Systems HMS 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896316321796

[7] M. Mendes, D. M. Pool, and M. M. Van Paassen, "Effects of peripheral visual cues in simulator-based training of multimodal control skills," 06 2017.

[8] D. M. Pool, G. A. Harder, and M. M. van Paassen, "Effects of simulator motion feedback on training of skill-based control behavior," *Journal of Guidance, Control, and Dynamics*, vol. 39, no. 4, pp. 889–902, 2016. [Online]. Available: https://doi.org/10.2514/1.G001603

[9] M. Mulder, D. M. Pool, D. A. Abbink, E. R. Boer, P. M. T. Zaal, F. M. Drop, K. van der El, and M. M. van Paassen, "Manual control cybernetics: State-of-the-art and current trends," *IEEE Transactions on Human-Machine Systems*, vol. 48, no. 5, pp. 468–485, 2018.

[10] M. Mulder, D. M. Pool, D. Abbink, E. Boer, and M. M. Van Paassen, "Fundamental issues in manual control cybernetics," vol. 49, 08 2016.

[11] P. Zaal and B. Sweet, "Estimation of time-varying pilot model parameters," 08 2011.

[12] R. Duarte, D. M. Pool, M. M. Van Paassen, and M. Mulder, "Experimental scheduling functions for global lpv human controller modeling," vol. 50, 07 2017, pp. 15 853–15 858.

[13] A. Popovici, P. Zaal, and D. M. Pool, *Dual Extended Kalman Filter for the Identification of Time-Varying Human Manual Control Behavior*. [Online]. Available: https://arc.aiaa.org/doi/abs/10.2514/6.2017-3666

[14] J. Rojer, D. M. Pool, M. M. van Paassen, and M. Mulder, "Ukf-based identification of time-varying manual control behaviour," *IFAC-PapersOnLine*, vol. 52, no. 19, pp. 109–114, 2019, 14th IFAC Symposium on Analysis, Design, and Evaluation of Human Machine Systems HMS 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S240589631931955X

[15] F. Tango and M. Botta, "Real-time detection system of driver distraction using machine learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 2, pp. 894–905, 2013.

[16] R. A. Skoog, T. C. Banwell, J. W. Gannett, S. F. Habiby, M. Pang, M. E. Rauch, and P. Toliver, "Automatic identification of impairments using support vector machine pattern classification on eye diagrams," *IEEE Photonics Technology Letters*, vol. 18, no. 22, pp. 2398–2400, 2006.

[17] A. Chaikheatisak and P. Chaiwuttisak, "Analysis of driver's attention through the internet of things (iots) for preventing road accident of natural gas vehicles," in *2021 7th International Conference on Engineering, Applied Sciences and Technology (ICEAST)*, 2021, pp. 168–172.

[18] S. K. R. Nittala, C. P. Elkin, J. M. Kiker, R. Meyer, J. Curro, A. K. Reiter, K. S. Xu, and V. K. Devabhaktuni, "Pilot skill level and workload prediction for sliding-scale autonomy," in *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 2018, pp. 1166–1173.

[19] R. Versteeg, "Classifying human control behaviour by artificial intelligence," Master's thesis, 2019. [Online]. Available: http://resolver.tudelft.nl/uuid:9adc6dfe-5f34-4936-a4d8-c0d35df7ee37

[20] G. Verkerk, "Classifying human manual control behavior in tracking tasks with various display types using the inceptiontime cnn," Master's thesis, 2021. [Online]. Available: http://resolver.tudelft.nl/uuid:74e19a71-01b9-4ce5-b684-15709771f1f7

[21] M. de Jong, "Classifying human pilot skill level using deep artificial neural networks," Master's thesis, 2021. [Online]. Available: http://resolver.tudelft.nl/uuid:f09da3e9-3220-46c4-9a3e-e066c9fb3ea0

[22] Z. Wang, W. Yan, and T. Oates, "Time series classification from scratch with deep neural networks: A strong baseline," 2016.

[23] A. P. Ruiz, M. Flynn, J. Large, M. Middlehurst, and A. Bagnall, "The great multivariate time series classification bake off: a review and experimental evaluation of recent algorithmic advances," *Data Mining and Knowledge Discovery*, vol. 35, pp. 401 – 449, 2020.

[24] S. Barendswaard, D. M. Pool, M. M. Van Paassen, and M. Mulder, "Dual-axis manual control: Performance degradation, axis asymmetry, crossfeed, and intermittency," *IEEE Transactions on Human-Machine Systems*, vol. 49, no. 2, pp. 113–125, 2019.

[25] K. Z. Six, "Rn-xcm: A neural network for classifying skill level in multi-axis tracking tasks," MSc. thesis, Delft University of Technology, Faculty of Aerospace Engineering, 2023, unpublished. [Online]. Available: http://repository.tudelft.nl

[26] R. Wijlens, P. Zaal, and D. M. Pool, "Retention of manual control skills in multi-axis tracking tasks," 01 2020.

[27] R. Stapleford, D. McRuer, and R. Magdaleno, "Pilot describing function measurements in a multiloop task," *IEEE Transactions on Human Factors in Electronics*, vol. HFE-8, no. 2, pp. 113–125, 1967.

[28] P. Zaal and D. M. Pool, *Multimodal Pilot Behavior in Multi-Axis Tracking Tasks with Time-Varying Motion Cueing Gains*. [Online]. Available: https://arc.aiaa.org/doi/abs/10.2514/6.2014-0810

[29] R. A. Hess, "Modeling human pilot adaptation to flight control anomalies and changing task demands," *Journal of Guidance, Control, and Dynamics*, vol. 39, no. 3, pp. 655–666, 2016. [Online]. Available: https://doi.org/10.2514/1.G001303

[30] G. A. Bekey, H. F. Meissinger, and R. E. Rose, "Mathematical models of human operators in simple two-axis manual control systems," *IEEE Transactions on Human Factors in Electronics*, vol. HFE-6, no. 1, pp. 42–52, 1965.

[31] E. Todosiev, "Human performance in a cross-coupled tracking system," *IEEE Transactions on Human Factors in Electronics*, vol. HFE-8, no. 3, pp. 210–217, 1967.

[32] W. Levison, J. Elkind, J. Ward, U. S. N. Aeronautics, S. Administration, and A. R. Center, *Studies of Multivariable Manual Control Systems: A Model for Task Interference*, ser. NASA CR. National Aeronautics and Space Administration, 1971. [Online]. Available: https://books.google.nl/books?id=sR6pxQEACAAJ

[33] Y. Iwai, M. Iwase, R. Ohno, and S. Hatakeyama, "Identification of human operation under closed-loop system," 12 2006, pp. 103 – 107.

[34] D. McRuer and H. Jex, "A review of quasi-linear pilot models," *IEEE Transactions on Human Factors in Electronics*, vol. HFE-8, no. 3, pp. 231–249, 1967.

[35] P. M. T. Zaal, D. M. Pool, J. de Bruin, M. Mulder, and M. M. van Paassen, "Use of pitch and heave motion cues in a pitch control task," *Journal of Guidance, Control, and Dynamics*, vol. 32, no. 2, pp. 366–377, 2009. [Online]. Available: https://doi.org/10.2514/1.39953

[36] M. Buda, A. Maki, and M. A. Mazurowski, "A systematic study of the class imbalance problem in convolutional neural networks," *Neural Networks*, vol. 106, pp. 249–259, oct 2018. [Online]. Available: https://doi.org/10.1016%2Fj.neunet.2018.07.011

[37] J. Johnson and T. Khoshgoftaar, "Survey on deep learning with class imbalance," *Journal of Big Data*, vol. 6, p. 27, 03 2019.

[38] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, *Efficient BackProp*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 9–48. [Online]. Available: https://doi.org/10.1007/978-3-642-35289-8_3

[39] V. Sharma, "A study on data scaling methods for machine learning," *International Journal for Global Academic Scientific Research*, vol. 1, 02 2022.

[40] P. P. Shinde and S. Shah, "A review of machine learning and deep learning applications," in *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*, 2018, pp. 1–6.

[41] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html

[42] I. J. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016, http://www.deeplearningbook.org.

[43] H. I. Fawaz, B. Lucas, G. Forestier, C. Pelletier, D. F. Schmidt, J. Weber, G. I. Webb, L. Idoumghar, P.-A. Muller, and F. Petitjean, "InceptionTime: Finding AlexNet for time series classification," *Data Mining and Knowledge Discovery*, vol. 34, no. 6, pp. 1936–1962, sep 2020. [Online]. Available: https://doi.org/10.1007%2Fs10618-020-00710-y

[44] F. Karim, S. Majumdar, H. Darabi, and S. Harford, "Multivariate LSTM-FCNs for time series classification," *Neural Networks*, vol. 116, pp. 237–245, aug 2019. [Online]. Available: https://doi.org/10.1016%2Fj.neunet.2019.04.014

[45] R. Assaf, I. Giurgiu, F. Bagehorn, and A. Schumann, "Mtex-cnn: Multivariate time series explanations for predictions with convolutional neural networks," in *2019 IEEE International Conference on Data Mining (ICDM)*, 2019, pp. 952–957.

[46] K. Fauvel, T. Lin, V. Masson, É. Fromont, and A. Termier, "XCM: An explainable convolutional neural network for multivariate time series classification," *Mathematics*, vol. 9, no. 23, p. 3137, dec 2021. [Online]. Available: https://doi.org/10.3390%2Fmath9233137

[47] A.-D. Pham, A. Kuestenmacher, and P. G. Ploeger, "Tsem: Temporally weighted spatiotemporal explainable neural network for multivariate time series," 2022. [Online]. Available: https://arxiv.org/abs/2205.13012

[48] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[49] I. Tougui, A. Jilbab, and J. E. Mhamdi, "Impact of the choice of cross-validation techniques on the results of machine learning-based diagnostic applications," *Healthcare Informatics Research*, vol. 27, no. 3, p. 189–199, 2021.

[50] S. Pouyanfar, S. Sadiq, Y. Yan, H. Tian, Y. Tao, M. P. Reyes, M.-L. Shyu, S.-C. Chen, and S. S. Iyengar, "A survey on deep learning: Algorithms, techniques, and applications," *ACM Comput. Surv.*, vol. 51, no. 5, sep 2018. [Online]. Available: https://doi.org/10.1145/3234150

[51] J. Patterson and A. Gibson, *Deep learning: A practitioners approach*. OReilly, 2017.

[52] F. Wang, J. Feng, Y. Zhao, X. Zhang, S. Zhang, and J. Han, "Joint activity recognition and indoor localization with wifi fingerprints," *IEEE Access*, vol. 7, pp. 80 058–80 068, 2019.

[53] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014. [Online]. Available: https://arxiv.org/abs/1409.1556

[54] S. Basodi, C. Ji, H. Zhang, and Y. Pan, "Gradient amplification: An efficient way to train deep neural networks," *Big Data Mining and Analytics*, vol. 3, no. 3, pp. 196–207, 2020.

[55] J. Bjorck, C. Gomes, B. Selman, and K. Q. Weinberger, "Understanding batch normalization," 2018.

[56] V. Nair and G. Hinton, "Rectified linear units improve restricted boltzmann machines vinod nair," vol. 27, 06 2010, pp. 807–814.

[57] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," 2014.

[58] M. Lin, Q. Chen, and S. Yan, "Network in network," 2014.

[59] Z. Guo, Q. Chen, G. Wu, Y. Xu, R. Shibasaki, and X. Shao, "Village building identification based on ensemble convolutional neural networks," *Sensors*, vol. 17, p. 2487, 10 2017.

[60] R. Naushad, T. Kaur, and E. Ghaderpour, "Deep transfer learning for land use and land cover classification: A comparative study," *Sensors*, vol. 21, p. 8083, 12 2021.

[61] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of Machine Learning Research*, vol. 13, no. 10, pp. 281–305, 2012. [Online]. Available: http://jmlr.org/papers/v13/bergstra12a.html

[62] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual explanations from deep networks via gradient-based localization," *International Journal of Computer Vision*, vol. 128, no. 2, pp. 336–359, oct 2019. [Online]. Available: https://doi.org/10.1007%2Fs11263-019-01228-7

[63] R. Fu, Q. Hu, X. Dong, Y. Guo, Y. Gao, and B. Li, "Axiom-based grad-cam: Towards accurate visualization and explanation of cnns," 2020.

[64] A. Chattopadhay, A. Sarkar, P. Howlader, and V. N. Balasubramanian, "Grad-CAM: Generalized gradient-based visual explanations for deep convolutional networks," in *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, mar 2018. [Online]. Available: https://doi.org/10.1109%2Fwacv.2018.00097

[65] D. V. Baelen, J. Ellerbroek, M. van Paassen, and M. Mulder, *Design of a Haptic Feedback System for Flight Envelope Protection*. [Online]. Available: https://arc.aiaa.org/doi/abs/10.2514/6.2018-0117

Part II

Preliminary Report

*This part has been assessed for the course AE4020 Literature Study.

3

Manual Control Cybernetics

This chapter presents background information regarding performed control cybernetics alongside a review of the current state-of-the-art research that has been in the field of cybernetics regarding both the acquisition of skill in human operators and the unique aspects introduced with the use of dual-axis tracking tasks. This chapter begins with Section 2.1 which introduces the topic of Cybernetics and how it models human control behaviour. Section 2.2 presents a literature study of the current cybernetic approach in assessing human operator skill levels and exploring phenomena found only in multi-axis tracking tasks. Lastly, Section 2.3 ends the chapter by summarising the shortcomings the cybernetic approach has when assessing human operator skill levels and the aspects a deep learning model would have to take into account when looking at data produced from a dual-axis tracking task.

3.1. Cybernetics

In 1983, Rasmussen [4] proposed that the decision-making done by humans that dictates their manual control behaviour can be distinguished into three distinct cognitive activities:

- Skill-Based Behaviour (SBB)
- Rule-Based Behaviour (RBB)
- Knowledge-Based Behaviour (KBB)

These behaviours, interrelation, and activating information can be seen in Figure 3.1.

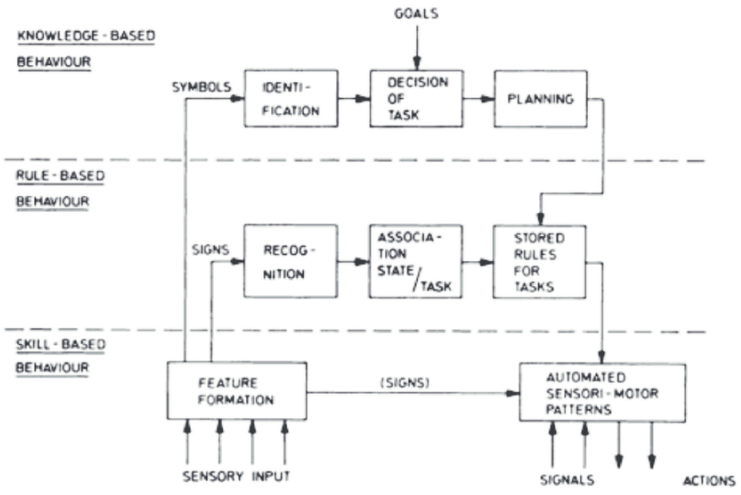

Figure 3.1: Skill-, Rule-, and Knowledge-Based Behaviour [4].

During manual control, humans simply interpret incoming sensory signals and send out actuating signals. A pilot that has little to no experience in executing a task would therefore cause them to not have the proper automated responses to accurately react to continuous signals transmitted by the task in hand

as would be found in a pilot who has greater experience. When looking at Figure 3.1, a pilot's manual control behaviour can be confined to the bottom right block of the figure as a closed-feedback loop between incoming signals and outgoing actions.

This analysis and the derivation of mathematical expressions for this closed-loop system is the field of Cybernetics. This analysis provides a basis for the rationalization and understanding of human control actions and in addition, would allow for predicting and understanding pilot-vehicle systems' behaviour.

3.1.1. Crossover Model

Due to the fact that humans as a controller are a non-linear and time-varying system, their behaviour cannot be explicitly linked to either mathematical or physical phenomena. In order to remove this non-linear and time-varying behaviour, McRuer proposed the *Crossover Model* [5]. This model splits the Human Operators' behaviour into a linear, time-invariant element, $H_p(s)$, and a non-linear, time-varying element called the remnant, n. This separation of Human Operator behaviour can be seen in Figure 3.2. It can be seen that this figure is essentially a modification of the block diagram of a compensatory tracking task as the Crossover model is applicable only to tasks using a compensatory display.

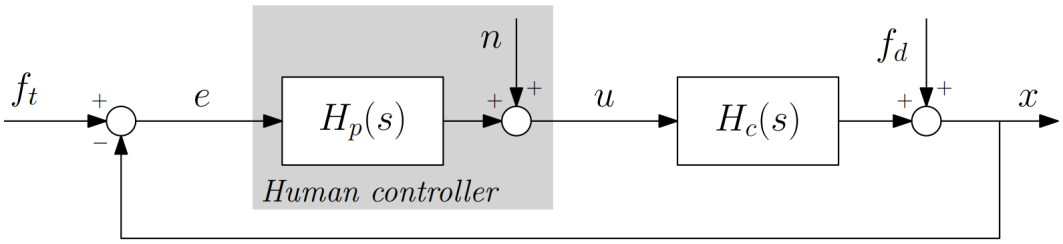

Figure 3.2: Block diagram of a compensatory tracking task [6].

McRuer determined that human operators would adjust their control behaviour based on several proposed 'Verbal Adjustment Rules'. Though there are many of these 'Verbal Adjustment Rules', the most significant one is 'equalisation'. This Verbal Adjustment Rule states that the human operators will always adjust their control behaviour based on the dynamics of the controlled element such that the entire system dynamics will become stable and behave as a single integrator at around ω_c, the crossover frequency. This is done such that the open loop transfer function around ω_c can be described by:

$$H_{OL}(j\omega) = H_p(j\omega)H_c(j\omega) = \frac{\omega_c}{j\omega}e^{-j\omega\tau_e} \tag{3.1}$$

In this equation, the $e^{-j\omega\tau_e}$ represents the innate time delay within humans that stems from the need to process and respond to the information displayed to them.

Once the human operator has stabilised the entire system into a single integrator system, the human operator uses the other 'Verbal Adjustment Rules' in order to minimise error and maximise performance. As $H_p(j\omega)$ represents the Human Operator, all the elements which they adjust in minimising the error are found within this variable. A comprehensive formulation of these elements is as follows:

$$H_p(j\omega) = \underbrace{K_p\left(\frac{T_L j\omega + 1}{T_I j\omega + 1}\right)}_{\text{equalisation}} \underbrace{\left(\frac{T_K j\omega + 1}{T'_K j\omega + 1}\right)}_{\text{low-freq. lag-lead}} \overbrace{e^{-j\omega\tau}}^{\text{delay}} \underbrace{\frac{1}{(T_N j\omega + 1)\left(\left[\frac{j\omega}{\omega_{nm}}\right]^2 + \frac{2\zeta_n j\omega}{\omega_{nm}} + 1\right)}}_{\text{neuromuscular dynamics, } H_{nm}} \tag{3.2}$$

Note that this formulation is specific only to a human operator's compensatory visual response. This is due to the fact that the gain of a human's visual sensor dynamics can be assumed to equal 1. For the case of a human's motion response, one needs to take into account the semicircular canal dynamics and its gain. This will be seen in Section 3.2.

Through the use of the crossover model, Equation 3.1, given that $H_c(j\omega)$, the dynamics of the system, is known, one can determine the human operator's elements or parameter values, thus successfully modelling a human's response to a control task [5, 7].

3.2. State of the Art Cybernetic Findings Relating to Research Topic

This section presents the current findings within the realm of cybernetics regarding assessing human operators' skill levels, the differences in human control behaviour when it comes to multi-axis tracking tasks compared to single-axis tracking tasks, and the objective evaluation of skill acquisition in dual-axis tracking tasks.

3.2.1. Cybernetic Approach in Assessing Skill Acquisition in Single-Axis Tracking Tasks

Pool and Zaal [1] utilised McRuer's approach of separating the Human Operator into a linear system and a non-linear system in order to assess a pilot's skill levels. From this, they recognised the multi-loop nature of the task at hand due to feedback from the human operator's visual and vestibular cues and thus, by using multi-channel models and different transfer functions for each of the perceived stimuli, were able to gain insight into the human operator's use of perceptual modalities when making a control action. The schematic representation of this approach can be seen in Figure 3.3.

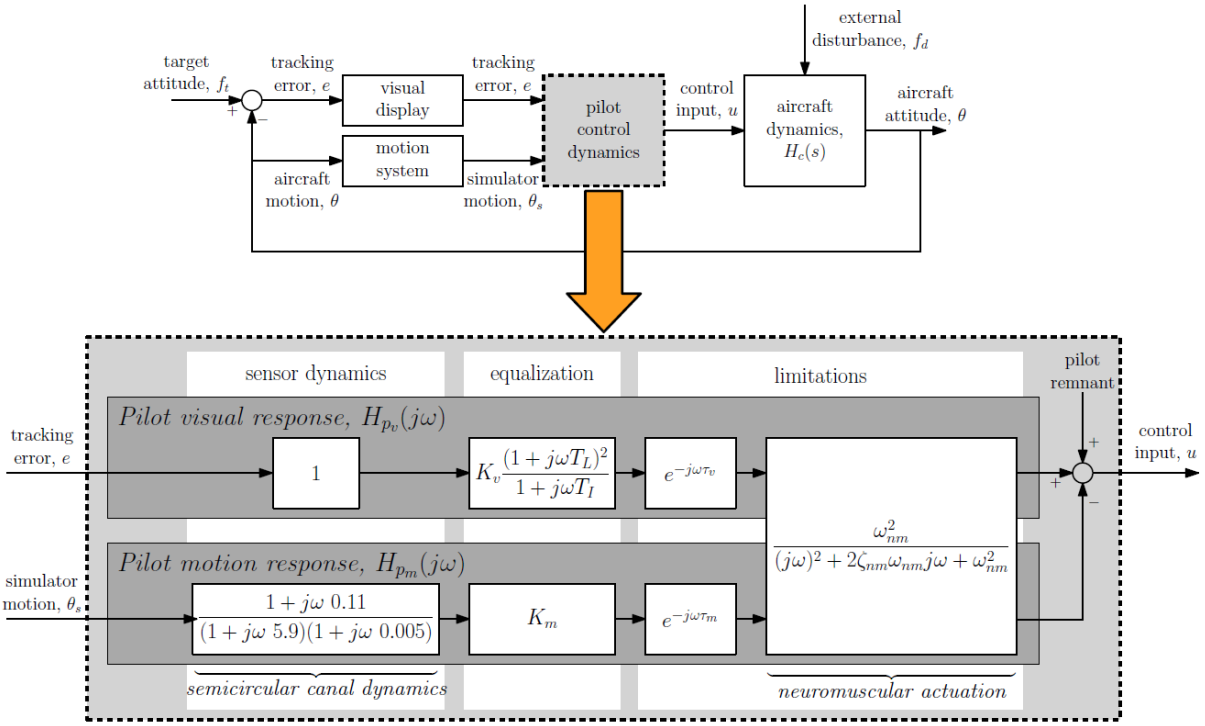

Figure 3.3: Schematic representation of the cybernetic approach in assessing skill level [1].

In order to use the cybernetic approach as shown in Figure 3.3 to quantify the human operator skill level, the models had to be fitted against measured human pilot data. Specifically, the free model parameters that characterize the human operator's control behaviour are as follows:

- K_v and K_m, the visual and motion gains
- T_L and T_I, the visual lead and lag time constants
- τ_v and τ_m, the visual and motion delays
- ω_{nm}, the neuromuscular system frequency
- ζ_{nm}, the neuromuscular damping ratio

By applying techniques that allow for the estimation of pilot model parameters from measured human operator data to individual tracking run data, Pool and Zaal [1] were able to obtain explicit quantification of human operators' utilisation of visual and motion cues in the training and transferring of manual control skills. The results of the experiment are shown in Figure 3.4.

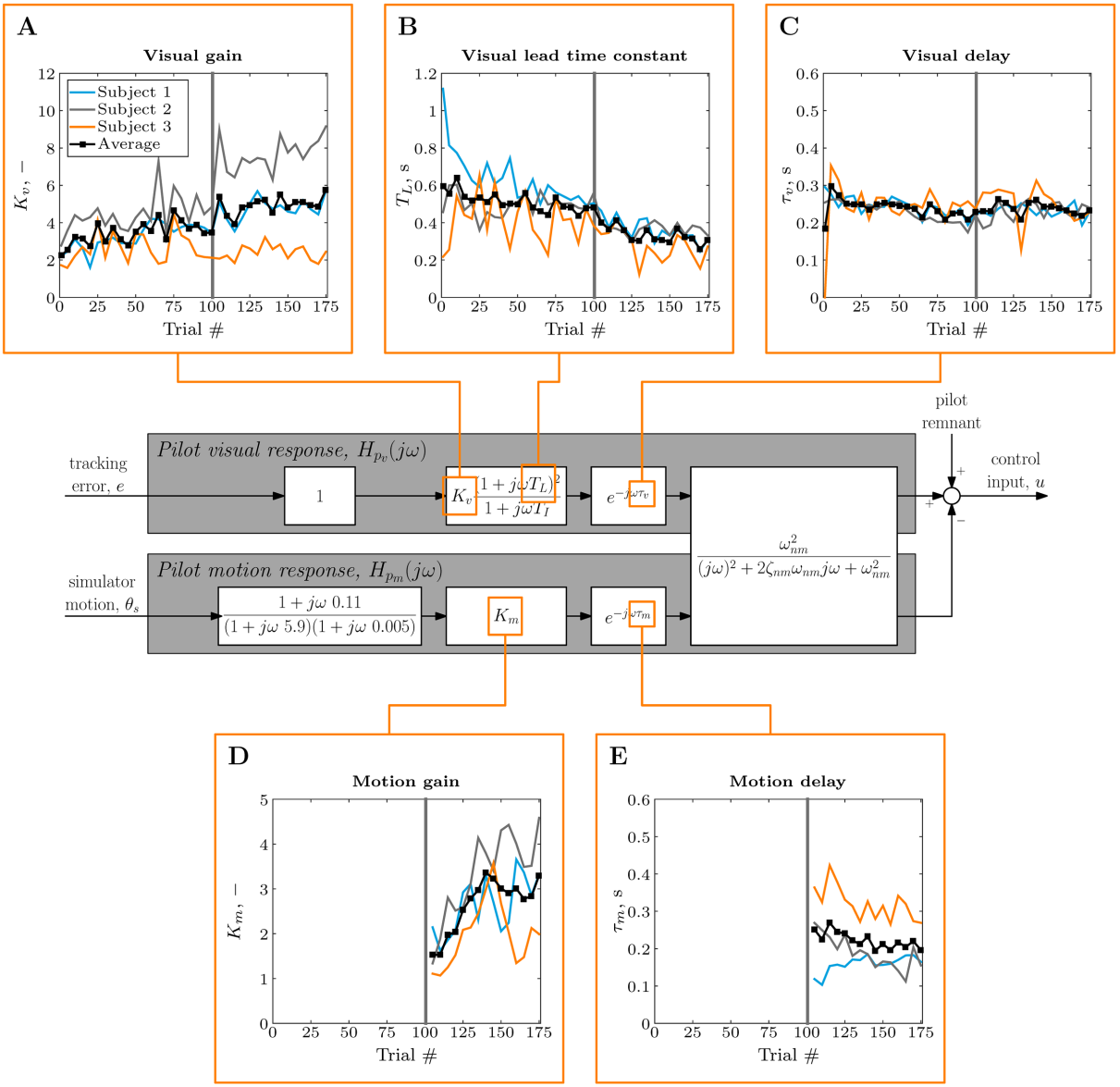

Figure 3.4: Progression of pilot model parameters [1].

Figure 3.4 presents the progression of the estimated human operator model parameters during the experiments. Note that the vertical line at trial 100 separates the inclusion of motion during the experiments. Specifically, prior to the line, no motion was used and after the line, motion was used. For the phase with no motion, it can be seen from graphs A, B, and C that with an increasing number of trials, and therefore with increasing skill levels, the human operators' K_v increases, T_L decreases, and τ_v decreases slightly respectively. When transferring to the phase with motion, it can be seen from the same graphs that the trends of the visual model parameters carry over even with the introduction of motion and carries on the same overall trend as those during the phase with no motion. It can also be seen from graphs D and E that with an increasing number of trails with motion, the human operators' K_m increases and τ_m not showing any consistent variation. Overall, it can be seen that a human operator's changes in their K_v, T_L, and

K_m are "highly consistent with the development of a multimodal control strategy" [1]. It was also found that motion is extremely important in the development of skill as the use of peripheral visual cues as a substitute for motion feedback proved to not be effective [8].

This cybernetic approach to assess the training of skill levels does come with a set of drawbacks. The first drawback stems from the separation of linear and non-linear aspects of the human operator's behaviour as this produces results that are linear and time-invariant. With humans being inherently non-linear and time variant, the results obtained by using the cybernetic approach do not provide a full picture regarding the human acquisition of skill. In addition, as the cybernetic approach requires the use of long time series measurements before obtaining meaningful results, this approach is unable to provide pilots feedback on their current skill level in real time. Another limitation which adds to the inability of providing real-time feedback is the inability of providing a fixed range of values of the human operator's model parameters which dictates whether or not there is a presence of skill acquisition. This is for the fact that every human operator displays a variety of values when it comes to their model parameters as can be clearly seen in Graph A. In this graph, it can be seen that Subject 1's K_v at trial 100 is within the same level as Subject 2's K_v at the early stages of their experiment. This puts on display the variability that exists between a human operator's model parameters and therefore implies that in order to be able to determine whether a person has indeed shown an increased level of skill, it is necessary to determine what values that specific person produces regarding their model parameter's when it comes to skill acquisition [1].

3.2.2. Time-Varying HO Parameter Estimation

Although it is true that most understanding of human manual control is limited to stationary and time-invariant tasks, progress has been made to understand better the adaptive nature of human control behaviour, i.e. the human's response to a sudden change in their environment, as currently, it is still not understood well [2, 7, 9, 10]. This aims to better understand human's adaptive nature however proves to be difficult due to the fact human adaptive nature is "highly variable, nonlinear, short-duration, and strongly task-dependent" [2], making it far more complex to understand than the better understood linear and time-invariant human control behaviour.

Literature on the adaptive nature of human control behaviour makes a distinction between two types of adaptations, 'fast' and 'slow' adaptations. The former refers to adaptations caused by sudden changes in tasks or environments [2] whereas the latter refers to adaptations caused by factors such as fatigue, loss-of-attention, and learning [9]. Although the understanding of human behaviour due to the 'slow' adaptations has been explored, as seen by the study done by Pool and Zaal [1] explored in the previous subsection, they do not fully capture the time-varying and non-linear nature of human control behaviour which does not hold when understanding human behaviour caused to 'fast' adaptations as they are truly non-linear and time-variant [2]. Therefore, methods that do allow for the full capture of such human control behaviour could allow for a better understanding of not only 'fast' adaptations, but also 'slow' adaptations such as skill acquisition.

In order to fully capture the non-linear and time-variant nature of human control behaviour, extensions of state-of-the-art methods are required. Several methods have already utilised for the identification of time-varying manual control behaviour such as generic maximum-likelihood estimation [11, 12], wavelets [12, 13], recursive least-squares [14, 15], Kalman filtering [16, 17, 18, 19], and a combination of both recursive least-squares and Kalman filtering [20]. Though all these methods show promising results, the biggest shortcomings that come with them is that they are suitable only for offline identification [21]. One method that shows potential for online identification is the use of recursive autoregressive ex-ogenous (ARX) models [21, 22].

Plaetinck et al. [21] utilised a previously developed recursive ARX identification technique [22] in order to allow for online estimation of the time traces of all the time-varying pilot model parameters using the measured tracking error signal e and the input signal u. In their research, Plaetinck et al. proposed two separate methods of utilising recursive ARX: the *Time-Invariant Condition Average* (TICA) method and the *Moving Average* (MA) method. The main difference between the two methods is the former's *a priori* explicit assumptions on the nature of the temporal variations whereas the latter makes no *a priori* assumptions. It was found that the TICA method was the most reliable method for identifying time-varying pilot adaptation with an accuracy of 57%. On the other hand, the MA method, the method which would be most practical for online application [21], yielded much lower accuracies at 15%. These low accuracies

suggest that at its current stage, modern methods for online identification of time-varying pilot behaviour are inadequate for accurate online identification of pilot skill level.

3.2.3. Dual-Axis Manual Control Behaviour

Although most relevant manual control tasks require the human operator to control systems of multiple degrees of freedom, most analysis and modelling of manual control tasks are limited to systems with a single axis. In fact, research that does tackle systems consisting of multiple degrees of freedom tackles the problem by separating them into multiple independent single-axis tasks. This approach does not take into account the inter-degree dependencies nature of the multi-axis task.

Barendswaard et al. [23] tackle this problem by investigating human control behaviour in dual-axis tracking tasks. By conducting a human-in-the-loop experiment in a moving-based simulator performing a dual-axis manual control task, they studied four distinctive phenomena that occur only in multi-axis systems:

- **Performance Degradation:**
  As evident from the name, Performance Degradation refers to the observed degradation of performance in dual-axis tasks when compared to single-axis tasks. Barendswaard et al. were able to observe the phenomenon in their experiments through the decreased crossover frequency stating that as the human operators distribute their attention to multiple axes, their performance in a singular axis decreases. They note that although the human operator's performance decrease, their control variance remains constant when compared to their single-axis tracking task performance. Furthermore, while studies suggest either a $1/\sqrt{2}$ ratio between the crossover frequency of dual-axis tracking tasks and single-axis tracking tasks [24, 25] or a relation stating that the crossover frequency is proportional to the reciprocal of the number of axes used [26], Barendswaard et al. did not observe such relations in their experiment.
- **Axis Asymmetry:**
  Axis Asymmetry refers to the distinct difference in the performance of each axis given equal controlled element dynamics in each axis. Barendswaard et al. discovered that although the control variance in the roll axis of the dual-axis control task was significantly higher than that of the pitch axis, participants displayed a consistently higher crossover frequency and lower error variance in the latter, therefore proving the presence of Axis Asymmetry. This preference for the pitch axis compared to the roll axis was also observed in earlier experiments [27, 28]. Barendswaard et al. theorised that this preference stems from the artificial horizon display used as pitch error was displayed with a resolution that was a factor of 2.3 pixels more than roll error per degree. It is also important to note that the addition of motion in the experiment improved performance, specifically lower error variances and higher stability, of both axes as a whole.
- **Crossfeed:**
  Crossfeed refers to the inability of the human operator to decouple the two tasks of dual-axis tracking. Put simply, Crossfeed occurs when a human operator's input in one axis feeds into another axis and thus affecting overall control performance. Previous research was able to observe the phenomenon but was unable to accurately capture the dynamics of Crossfeed in the frequency range. Barendswaard et al., however, were able to successfully detect, identify, and model the phenomenon using a novel extended Fourier coefficient method [27, 29]. It was found that Crossfeed has a significant 20% contribution to the measurement of human operator control inputs and that the addition of Crossfeed improved the modelling accuracy of multi-axis control by 5% suggesting that Crossfeed is a key attribute to human multi-axis control.
- **Intermittency:**
  Intermittency refers to the time-varying axis prioritization of human operators in multi-axis control tasks. Due to its time-varying nature, it has been modelled as a pilot remnant and has been observed to proportionally increase with each additional axis. This phenomenon was noted by previous experiments to occur when one axis has a larger error than the other which results in temporary prioritization over the other. Barendswaard et al. discovered the presence of this phenomenon through the observation of peak time differences of the roll axis in a dual axis compared to the roll axis in a single axis. Although they theorise that Intermittency is caused by the display of the signal's perceivability, they recommend further investigation regarding the cause of this phenomenon.

3.2.4. Objective Evaluation of Skill Acquisition in Dual-Axis Tracking Tasks

Wijlens et al. [30] conducted a study to objectively and quantitatively evaluate the acquisition, decay, and retention of skill in compensatory dual-axis tracking tasks. As this research, is mainly focused on the acquisition of skill, or the detection of someone with and without skill, Wijlens et al. results regarding this matter will be looked at.

As can be seen in Figure 3.5, tracking error is consistently lower in the pitch axis than in the roll axis during the training phase which is in line with Barendswaard et al.'s findings regarding Axis Asymmetry [23]. Additionally, it was found that in the pitch axis, participants displayed higher learning rates and earlier stabilisation during training and a lower rate of decay in performance after a significant period of time with no practice.

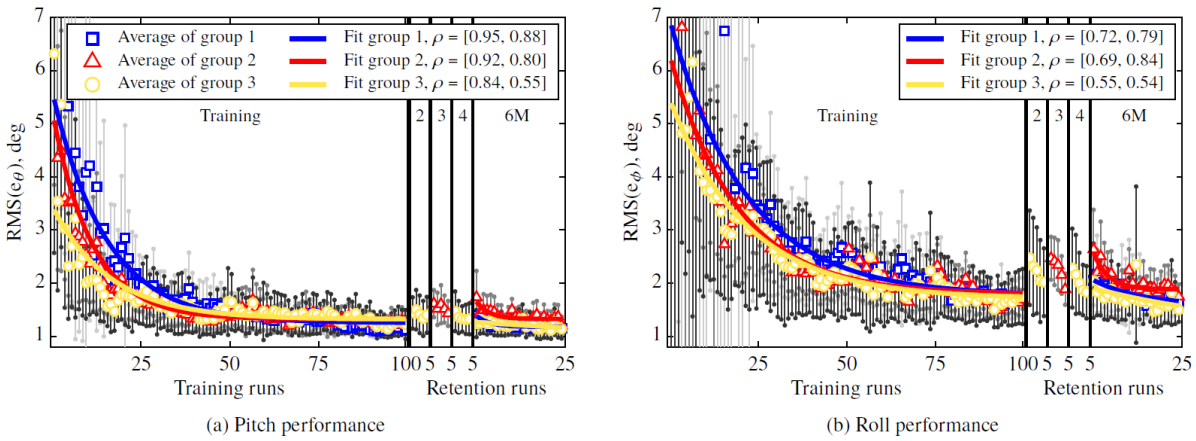

Figure 3.5: Average pitch and roll tracking error with increasing runs [30].

As it was found by Barendswaard et al. that Crossfeed has a significant contribution to human operator control inputs, 20% to be exact [23], Wijlens et al. attempted to verify this in their study. Wijlens et al.'s findings are shown in Figure 3.6 and Figure 3.7. Both figures display the contribution from the target signal of the principal axis, Crossfeed, and remnant to tracking error variance and control input variance respectively. As can be seen in Figure 3.7, Crossfeed's contribution to measured control inputs was only up to 8%, much lower than the claimed 20% with more Crossfeed contribution in the roll axis than the pitch axis. When looking at both figures, Crossfeed's contribution to both skill and control input was constant with an increase in runs. Rather, it can be seen in both figures that consistently, the remnant had more contributions. Specifically, with lower training runs, the remnant had extremely high contributions which gradually lowers with higher runs. The previous findings of Axis Asymmetry are further proven by these two figures where the decrease of remnant contribution in the pitch axis, the left-hand side of the two figures, is far steeper than those found in the roll axis, the right-hand side of the two figures. Wijlens et al. determined that the lower contribution of Crossfeed is attributed to the differences in manipulator location in the conducted experiments. However, overall, it can be seen that remnant is a signal that is extremely important in the detection of skill [31].

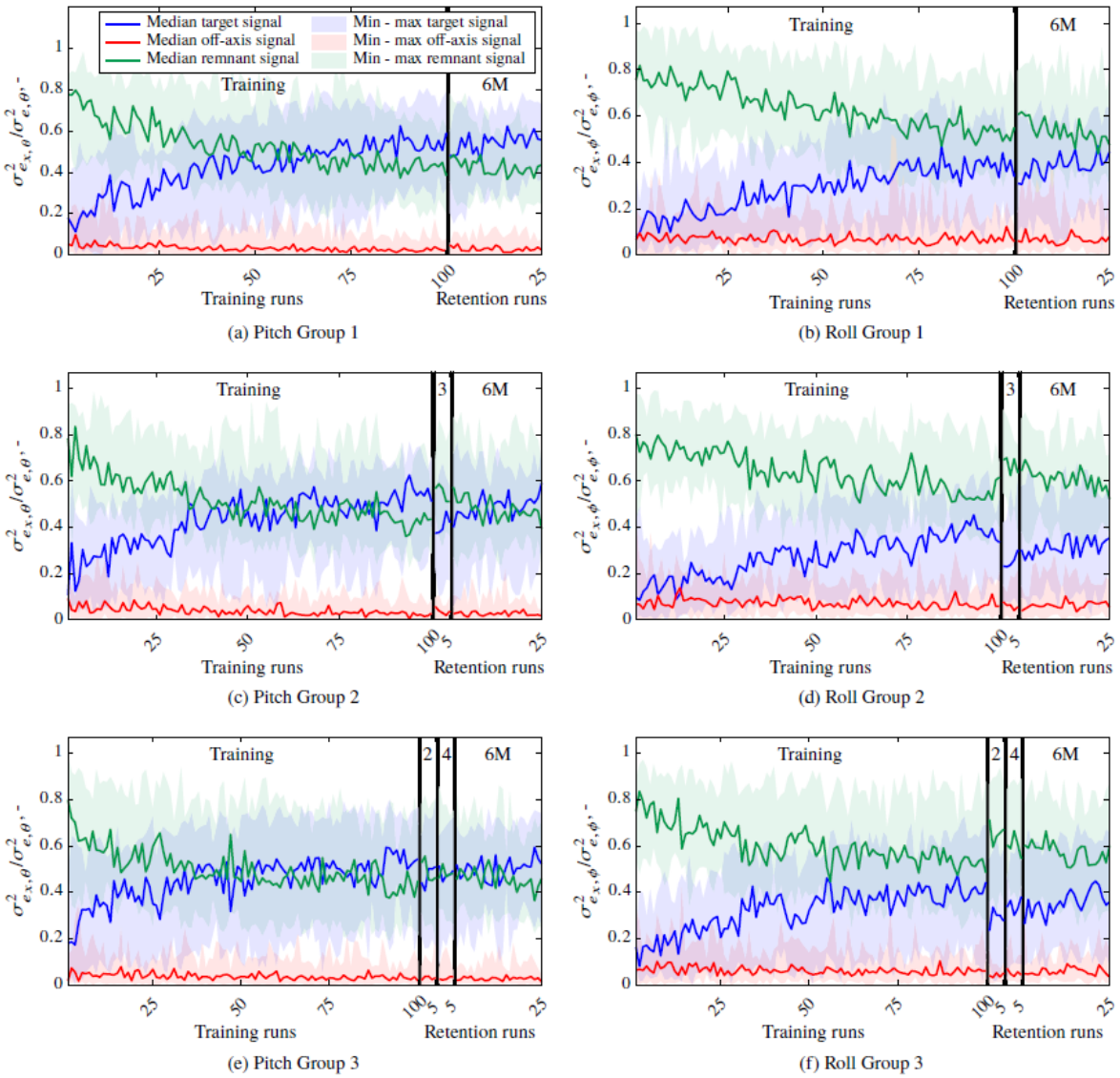

Figure 3.6: Normalised principal-axis target, off-axis target, and remnant signal contributions to tracking error variance [30].

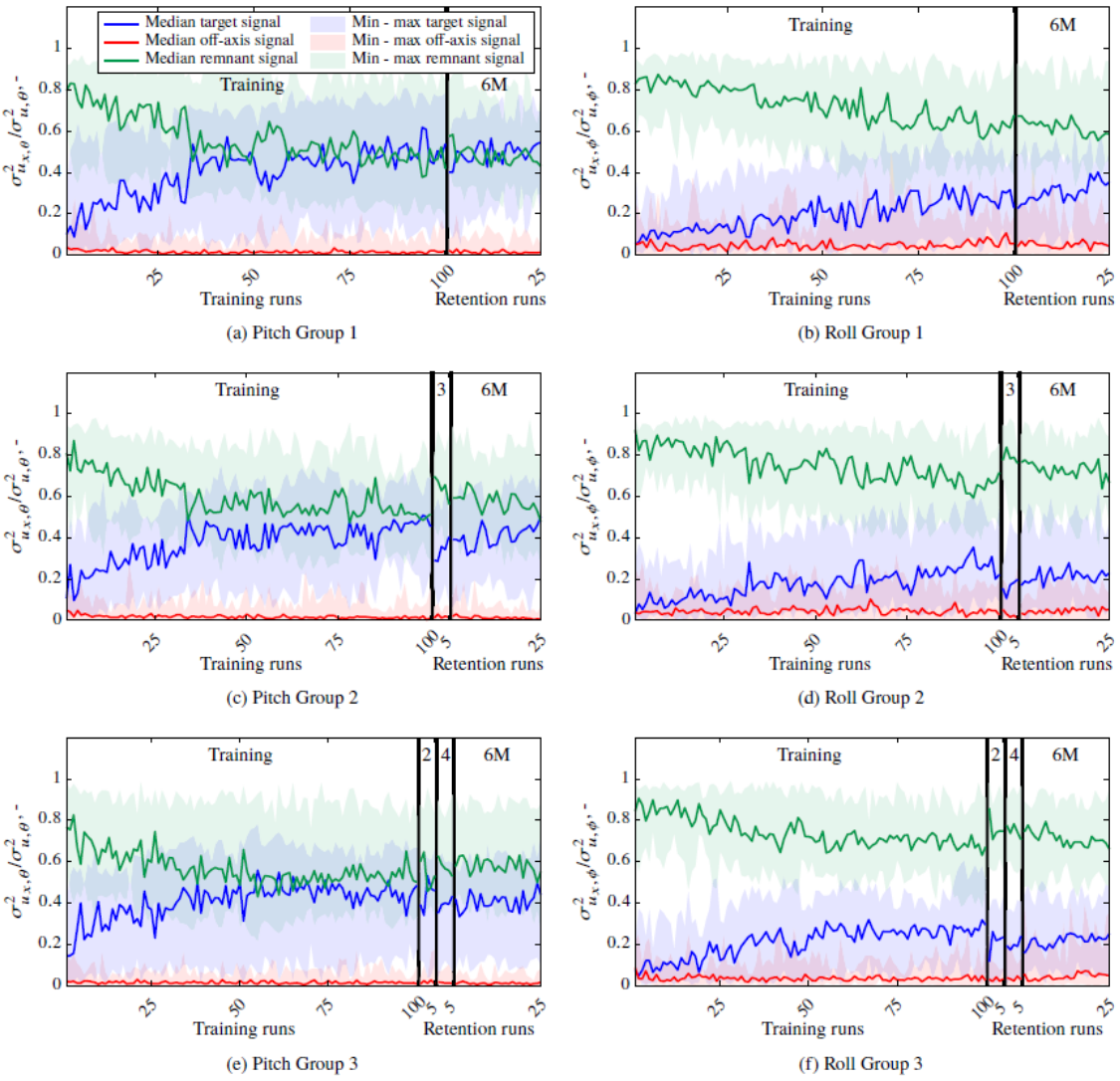

Figure 3.7: Normalised principal-axis target, off-axis target, and remnant signal contributions to control input variance [30].

3.3. Key Takeaways

This chapter provides background information on the current standards of human modelling techniques in Cybernetics, as well as research in the field on assessing human operator skill levels. It also investigates the phenomenon introduced in multi-axis control tasks as well as objective observations in the acquisition of skills in dual-axis tracking tasks. The key takeaways of this chapter are as follows:

- The cybernetic approach in assessing human skill levels, though effective, brings with it a few drawbacks. The biggest drawback is the fact that it produces results that are linear and time-invariant due to the nature of the crossover model. In addition, to produce meaningful results, a large set of long time series data for each human operator is required. This removes any possibility of providing real-time feedback for any human operator.
- Research has been conducted in order to better understand the time-varying nature of HO control behaviour however most modern attempts at identifying these time-varying behaviour are applicable only to offline usage. Methods that do allow for online usage are currently not accurate enough to be used reliably.
- The introduction of multi-axis tracking tasks adds four distinct phenomena that affect the human

operator's proficiency in the task. Therefore, it would be of great interest to determine whether the developed model to assess HO skill level would be able to detect these phenomena as it could either have a positive or negative impact on the model's ability to classify skill level.

- When objectively evaluating and quantifying the acquisition of skill in dual-axis tracking tasks, it can be seen that there indeed exists the presence of Axis Asymmetry however, Crossfeed was observed to have much lower contributions overall than previously believed due to differences in manipulator locations. Instead, it was found that the remnant signal had the most contribution with significant contribution in earlier runs, or lower skills, and lower contributions in later runs, or higher skills. Therefore, the model would need to be able to identify the non-linear and time-varying signal that is the remnant.

This research aims to provide a deep learning model which would be able to provide a pilot with real-time feedback regarding their skill level in real-time. Therefore, the model should be able to identify the non-linear and time-varying aspects of human manual control behaviour, the remnant, without the need for both long-time sequences and the need to train on a person's specific data in order to make any judgement in skill level. In addition, the deep learning model should be able to delve and look into the interdimensional relationship that occurs in multiaxis tracking data to be able to recognise the effects the four distinct phenomena that were stated by Barendswaard et al. have on performance while keeping an emphasis on the remnant signal.

4

Neural Networks

In recent years, Deep Learning, or put simply, the use of NNs, have become extremely popular. The most popular recent example of this is the popularity and usefulness of ChatGPT [32]. The chapter aims to explain the main idea behind NNs, the most popular NN architectures within the task of MTSC, a review of state-of-the-art knowledge regarding classifying skill level in HO control data, the selection of potential NN models to use in this research, and the possibility of applying explainability in AI.

This chapter begins by introducing how simple NNs work in Section 3.1. Following this, Section 3.2 presents the two major NN architectures which are predominantly used in MTSC alongside an explanation of how each works. Section 3.3 then presents the current understanding of using NNs to classify the skill level of human control behaviour. Section 3.4 then presents the list of NN models which will be used in the preliminary phase of this research alongside the criteria which was used to select them. Lastly, Section 3.5 present the possible methods which could be used to add an explainability aspect to the NN model with one being chosen to be used for the rest of the research.

4.1. Inner Workings of Neural Networks

Before delving too deep into the world of NNs, it is best first to explain their working principles. This is best done using the simplest form of a NN, known as either a Feedforward Neural Network (FNN) or a Multilayer Perceptron (MLP).

4.1.1. Structure of Neural Networks

As suggested by the name, NNs are inspired by the brain's ability to learn, adapt, or make decisions through a collection of interconnected neurons that send signals to one another [33]. This inspiration can be clearly seen in Figure 4.1 where each circle, also called the node or the neuron, represents the brain's neurons and the lines between each one of these nodes represent the path with which information is sent from one brain neuron to the other. It can also be seen that the FNN shown in Figure 4.1 is separated into three distinct layers: the input layer i, the hidden layer h, and the output layer o. In this example, the FNN contains three hidden layers as represented by h_1, h_2, and h_3. The number of hidden layers and nodes within these layers are hyperparameters with which the model architect optimises in order to produce the best results out of their current problem. The number of nodes within the input and output layer however depends specifically on the task at hand.

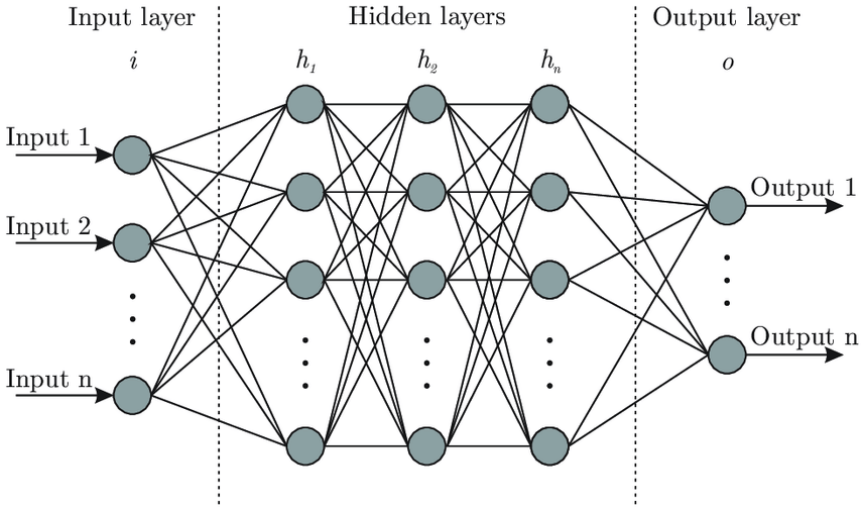

Figure 4.1: Network structure of an FNN [34].

Starting from the input layer, the nodes in this layer take an input value, x, and essentially hold it. The working mechanism of NNs begins with the connection between the input layer and the first hidden layer, in the case of Figure 4.1, h_1. The input layer then sends its information to a node from the hidden layer where it is multiplied by a weight, w. Note that each connection has its own unique weight. After being multiplied by the weight, a bias, b, is then added to it. Each node has its own unique biases except those found in the input layer. With the value from the input node being multiplied by weight w and summed by bias b, it is used as an input for the receiving node's activation function, ϕ, to obtain the final activation value, a, of the neuron which will act as the input for the nodes found in the next hidden layer, repeating the process again [33]. Note that this activation function is not unique to one node in the hidden layers as they all share the same activation functions. A mathematical representation of this process can be seen as follows:

$$\mathbf{a} = \phi(\mathbf{W} \cdot \mathbf{x} + \mathbf{b}) \tag{4.1}$$

In this equation, **a** represents the vector containing all the activation values of each node in a single hidden layer, **W** represents the matrix containing all the weight between each node from the input and hidden layer, **x** represents the vector containing the input values from the nodes in the input layer, and **b** represents the vector containing all the biases of each node in the hidden layer. A visual illustration of this mathematical expression for a single node can be seen in Figure 4.2 where a sigmoid function is used as the node's activation function ϕ.

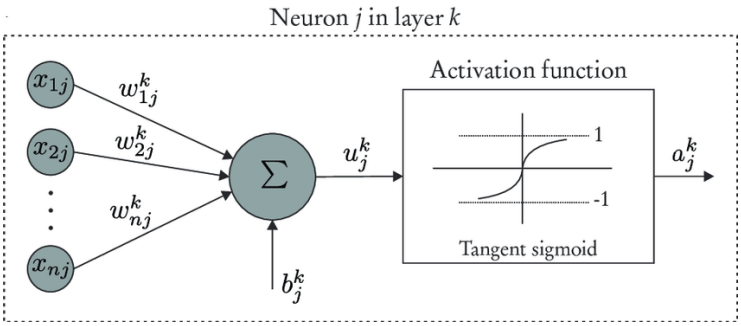

Figure 4.2: Neuron j in layer k obtaining its activation value a [34].

Looking deeper into the activation function, ϕ, this is essentially a function that prevents NNs from being a linear regression model as can be seen in Function 3.1 if ϕ were to be removed. Therefore, activation

functions are the reason why NNs are able to learn the non-linearities of their inputs [33]. Note that there are many different activation functions with the three most popular ones being: the sigmoid function, as shown in Figure 4.2, the hyperbolic tangent (tanh) function, and the Rectified Linear Unit (ReLU) function [33]. Each function has its different use case which the model architect must evaluate to obtain the most suitable one for the model's purpose.

Once the above process is repeated through all the hidden layers, the activation values will need to be transferred to the output layer. The process of doing is exactly the same as the process for the hidden layers with the exception that instead of creating an activation value, a, it creates an output value, $\hat{y}$, and takes in said activation value as the input. A mathematical representation of this can be seen as follows:

$$\hat{\boldsymbol{y}} = \phi(\mathbf{W} \cdot \mathbf{a} + \mathbf{b}) \tag{4.2}$$

In this equation, $\hat{\boldsymbol{y}}$ represents the vector containing all the output values from the nodes in the output layer.

The biggest difference between the process of obtaining output values and the process of obtaining activation values is the choice of activation functions [35]. In a classification task, such as the one in this research, two main activation functions exist, the sigmoid function and the softmax function. Sigmoid functions are applicable only to binary classification problems whereas softmax functions are applicable to both binary and multi-class classification problems. After applying the activation function, each node's output value is essentially the percentage with which the system believes the data is classified with the highest value being the system's 'choice' in the classification task. Evidently, the sum of all the output values should equal 100%.

4.1.2. Training of Simple Neural Network

With the manner in which NNs work established, it is now necessary to explain how NNs do their most important task, learning. Before data has even entered the NN, its weights and biases are assigned pseudo-randomly at its initialisation. Then, training data is fed to the NN in 'batches' with the purpose of improving the efficiency of training and making training more computationally feasible. The batch size, the number of samples within a batch, is another hyperparameter that the model architect optimises. Once the first batch of training data is processed by the NN, it will return its output $\hat{\boldsymbol{y}}$ which is the model's estimate of the actual and correct values $\mathbf{y}$.

With the model's initial predictions obtained, the actual training of the NN begins using the backpropagation algorithm [35]. In this algorithm, both $\hat{\boldsymbol{y}}$ and $\mathbf{y}$ are passed through the loss function. The loss function is an important function as it is the function that returns a number representing the performance of the NN, or in simpler terms, how 'wrong' the system is [33]. Therefore, the goal of the NN is to minimise the loss function. The use of a loss function in order to determine the performance of the system compared to the usage of the number of incorrect guesses, i.e. $\boldsymbol{\epsilon} = \hat{\boldsymbol{y}} - \mathbf{y}$, is preferred due to two major reasons [35]:

- The loss function's ability to provide a continuous and differentiable measure of the error of the NN's prediction.
- The loss function's ability to provide the severity of each mistake.

Note that there are many different types of loss functions that the model architects could decide to use however, it is in their best interest to select the one most appropriate for the task at hand.

The error produced by the loss function is then propagated backwards, hence the name of the algorithm, through the NN from the output layer to the input layer. The error is then used by the 'optimiser' to compute the gradient of the cost function with respect to the weights and the biases which tell the NN the direction in which the weights and biases should be adjusted in order to minimise the loss function. Similar to the loss function, there exist many types of optimisers and the model architect selects the most appropriate one. The magnitude in which the weights and biases should be adjusted is determined by the 'learning factor' or 'learning rate' which is a hyperparameter that the model architect also optimises. This is due to the fact that if the learning rate is too low, the model will take too long to reach its minimum whereas if it is too high, the model will update too drastically to produce any convergence and may instead begin to diverge away from the minimum. This can be seen in Figure 4.3.

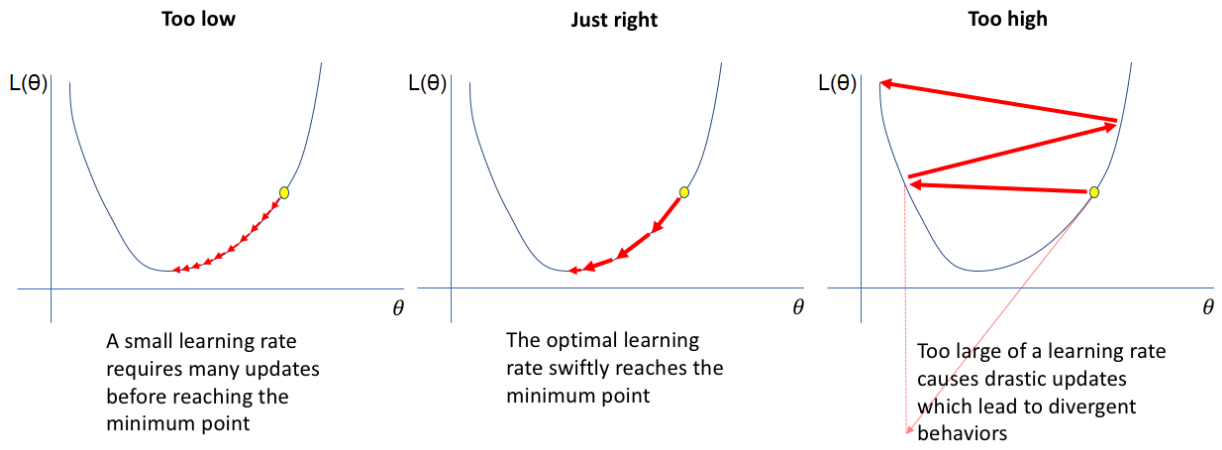

Figure 4.3: The effects different values of learning rate have on minimizing loss [36].

Once the weights and biases have been updated, the next batch of training data is then passed through the NN and the process is repeated. Once the entire training data has been passed through the NN, the validation data is then passed through the model. The validation data is unique in that this data is not used to train the NN model and optimise its parameters but instead, is used to evaluate the model after its training and allow for the detection of overfitting, a concept that will be discussed further in the next subsection. Thus it is important to note that results from the validation data will not be used to conduct any backpropagation. Once the model has gone through both the training and validation data, the model repeats the whole process again for the number of iterations, or epochs, specified by the model architect. It is very important to note that due to the stochastic nature of the training phase, the system is not guaranteed to reach a global minimum in minimising the loss function [33]. Therefore, model architects always train their networks multiple times in order to determine the overall average performance of their model.

4.1.3. Overfitting in Neural Networks

One of the biggest drawbacks of NNs is the presence of overfitting due to their large number of trainable parameters [33]. Overfitting can be seen as the NN training too well with the training data such that it memorizes the data rather than the underlying pattern [33]. This results in a continued increase in performance with the training data, but a decrease in performance with validation data. This can be seen in Figure 4.4.

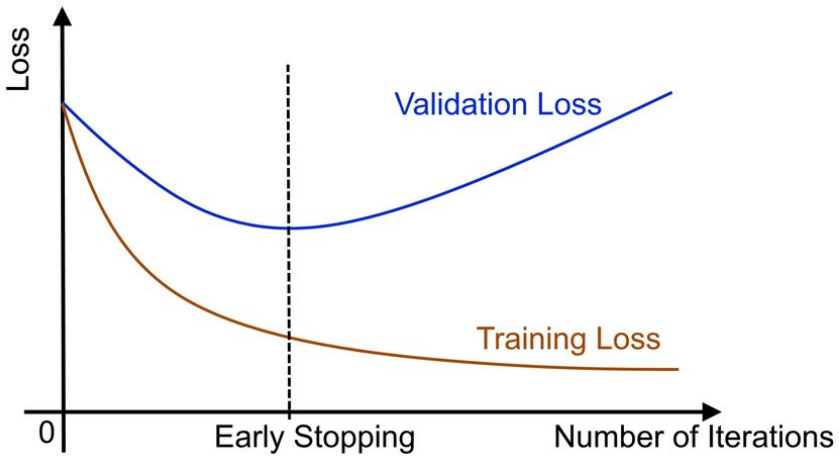

Figure 4.4: Visual illustration of a NN model overfitting alongside its recommended Early Stopping point [37].

There are numerous techniques to overcome overfitting in NNs with a few listed below:

- Early Stopping [33]:
  As illustrated in Figure 4.4, Early Stopping involves stopping the training of the network when the loss of the validation data begins to increase while the loss of the training data continues to decrease.
- Dropout [38]:
  Dropout involves 'dropping out' a certain number of nodes, in other words turning them off, represented by a Dropout rate. For example, if a Dropout rate of 25% is chosen by the model architect, 25% of the nodes are turned off which are randomly chosen during each epoch and their weights will not be updated. This reduces the possibility of the neuron from co-adapting with one another too much. Note that dropout only applies during training and that during evaluation with the validation data, dropout will not be applied.
- Data Augmentation [39]:
  One of the major reasons NN models tend to overfit is the lack of abundance in training data. One way to overcome this is by augmenting the training data to create new datasets. There are numerous methods used with the goal of data augmentation but they are beyond the scope of this research.

4.2. Major Neural Network Architectures used for Multivariate Time Series Classification

In looking at the majority of models designed for MTSC [40, 41, 42, 43, 44, 45, 46], two NN archetypes are seen to be used frequently: Convolutional Neural Networks and Recurrent Neural Networks. This chapter aims to explain the inner mechanisms of these two archetypes to allow for an understanding of their frequent use in MTSC.

4.2.1. Convolutional Neural Networks

Convolutional Neural Networks (CNN) are a specialised type of NN that emerged due to the limitations regular NNs faced when it came to image recognition [35, 47, 48]. Specifically, regular NNs faced the following problems:

- No awareness of spatial geometry [47]
- Requires the use of many connections [35]
- Prone to overfitting [35, 48]

CNNs overcome the limitations that arise from the use of regular NNs by copying the visual cortex, the part of the brain responsible for processing visual information [35, 47]. Essentially, the visual cortex processes visual information hierarchically in a series of stages where each stage builds on the representations learned in the previous stage. This is copied into CNNs by having Convolutional layers.

The Convolutional Layer, which effectively replaces the hidden layer found in FNNs, is the stage of neurons that learns representations found within an image. They do so through the use of kernels or filters, with convolutional layers consisting of multiple kernels promptly named 'Multi-Channel Convolutional Layers'. The number and size of these kernels are hyperparameters optimised by the model architect. These kernels essentially slide across the image shown to the network, applying its trainable weights and bias to each snippet of the image to produce a feature map [33]. An example of a kernel sliding across input data can be seen in Figure 4.5.

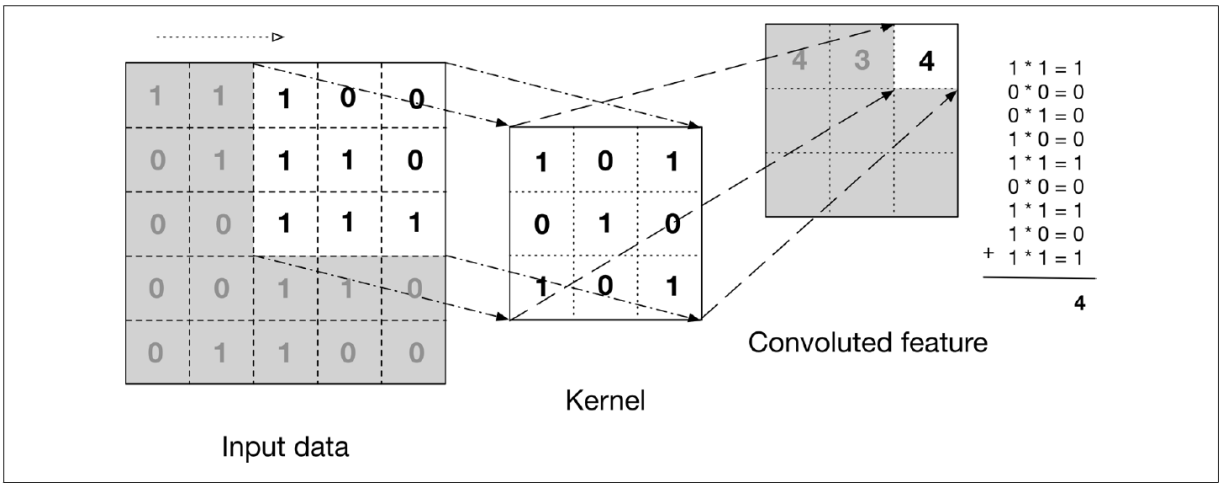

Figure 4.5: A kernel sliding across input data to create a feature map [33].

The weights of each kernel are the learnable parameters that extract the features out of the image whereas the biases of each kernel are the learnable parameters that signify the importance of the features extracted from the kernel. Note that each kernel in a convolutional layer produces its own feature map. Similar to the hidden layers found in the FNNs, these feature maps are passed through activation functions to allow the learning of non-linear relationships within the data. Within CNNs, it is expected to have multiple convolutional layers such that the earlier layers will recognise low-level features such as edges such that the following convolutional layers will combine these lower-level features to recognise high-level features such as faces. An example of a CNN recognising these lower and higher-level features can be seen in Figure 4.6.

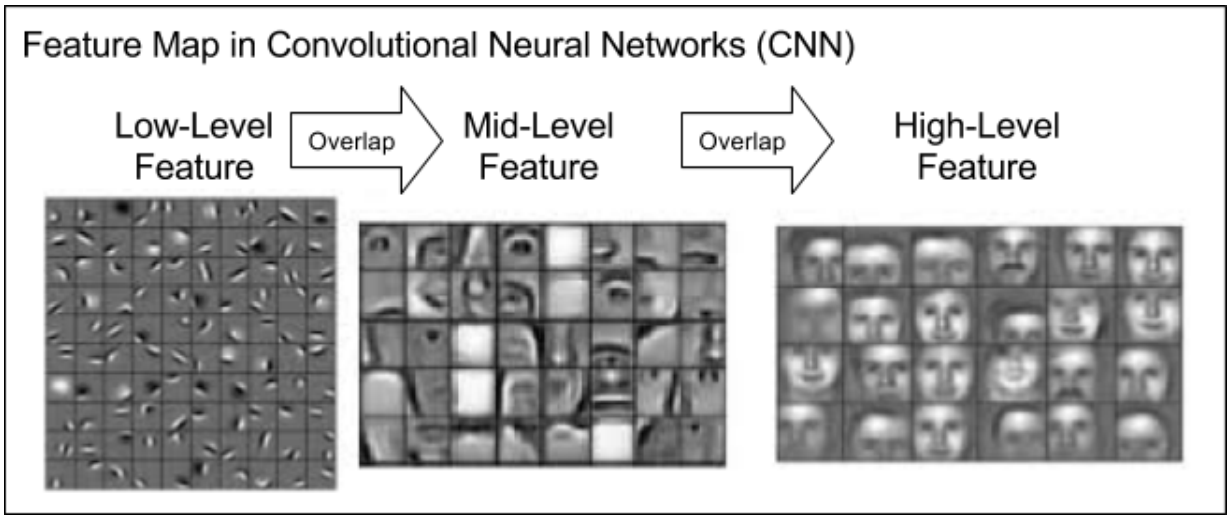

Figure 4.6: Feature mapping at different levels of a CNN in facial recognition [49].

Once the convolutional layer has produced its feature maps, they are then used as inputs for the Pooling layer. The pooling layer is another layer that is unique to CNNs as their purpose is to downsample and reduce the spatial dimensions of the feature maps produced in order to reduce computational load, increase efficiency, and more importantly, capture the important information found within these features maps while suppressing the irrelevant information, thus reducing overfitting [48, 50]. Similar to activation functions, loss functions, and optimisers, there are many different kinds of pooling functions that allow for the pooling of feature maps which the model architect determines the optimal one for the task at hand. It is important to note although they are layers, these pooling layers contain no trainable parameters as they are simply functions.

The outputs of the pooling layers are then passed through a fully-connected layer, similar to a hidden layer found in FNNs, in order to analyse the downsampled feature maps and pass the information through a softmax layer in order to make a classification guess [33]. The training of CNNs then begins from here using the same backpropagation algorithm explained earlier for FNNs. A visual illustration of the flow of information in CNNs can be seen in Figure 4.7 where a CNN with two multi-channel convolutional layers analyse a street sign.

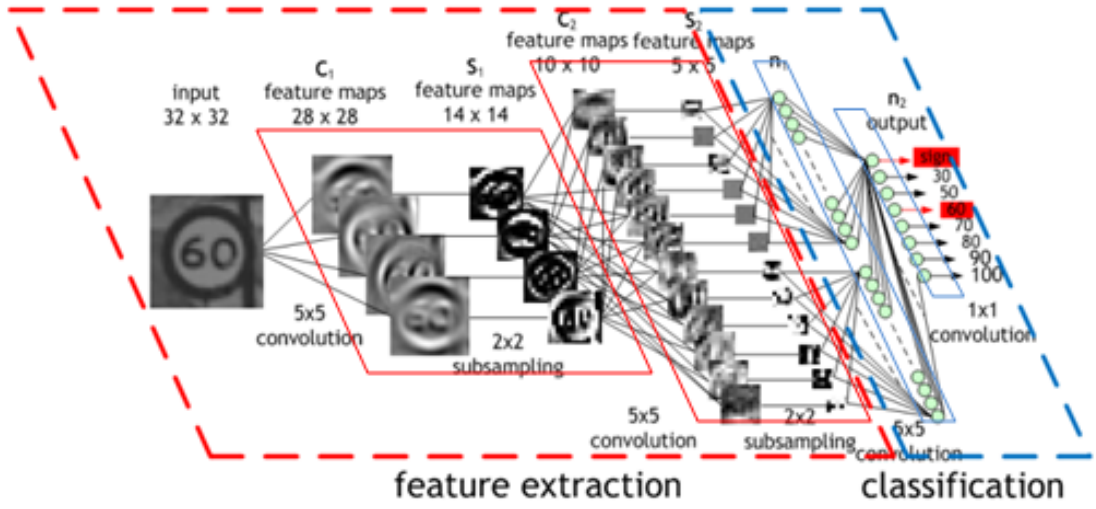

Figure 4.7: Steps taken by a CNN with two multi-channel convolutional layers in analysing an image [51].

4.2.2. Recurrent Neural Networks

Recurrent Neural Networks (RNN) are a specialised type of NNs that emerged to model sequential data. RNNs work by processing a sequence of inputs one at a time while maintaining sequential memory, an internal state that summarizes the input seen before [50].

An illustrative example of how RNNs work can be seen in Figure 4.8 where x represents the input, h represents the hidden state, and o represents the output. In this left section of the figure, RNNs' most distinctive feature can be seen in the form of an arrow encircling h. This arrow showcases an RNN's ability to 'remember' the previous h. This can be better explained by 'unfolding' the figure such is showcased on the right section of the figure. Here, it can be clearly seen how the past input of a sequence, x_{t-1}, is used to calculate the previous hidden state, h_{t-1}. This previous hidden state is then used alongside the current input of the sequence, x, as an input for the current hidden state h. The hidden states h is also used to create outputs of the model o at each time step. Although it is not explicitly illustrated in Figure 4.8, RNNs also contain biases alongside weights which are illustrated as V, W, and U.

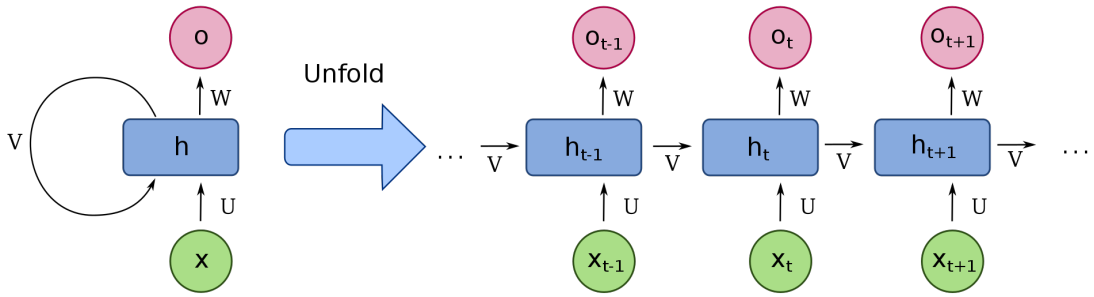

Figure 4.8: General structure of an RNN [52].

Note that although both o_{t-1} and h_t use h_{t-1} as inputs, they are not equal due to the fact that they each have their own weights and biases alongside the fact that h_t also takes in x_t as input alongside its own weight and bias. This is mathematically expressed as where ϕ represents the model architect's chosen activation function:

$$h_t = \phi_h(U \cdot x_t + V \cdot h_{t-1} + b_h) \text{ where, } h_{t-1} = \phi_h(U \cdot x_{t-1} + V \cdot h_{t-2} + b_h) \tag{4.3}$$

$$y_t = \phi_y(o_t) = \phi_y(W \cdot h_t + b_o) \tag{4.4}$$

A deeper look into the RNN's cell which takes into account these mathematical expressions can be seen in Figure 4.9 with a *tanh* function being used as the hidden state's activation function ϕ_h.

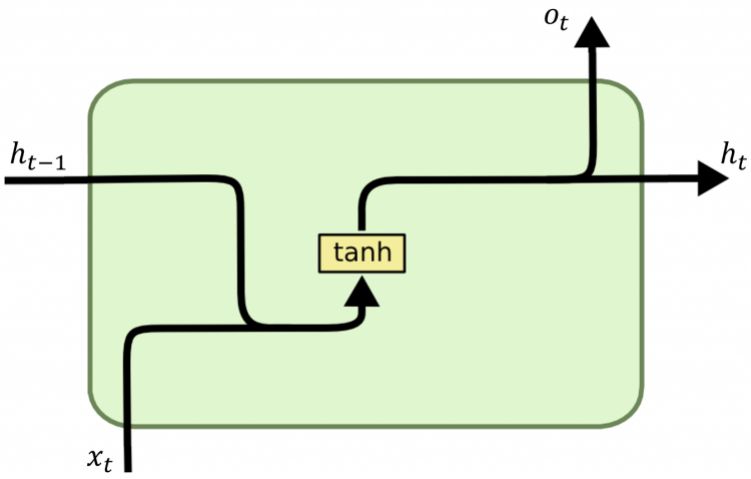

Figure 4.9: Flow of information in an RNN cell [53].

The biggest drawback that comes with RNN's recursive nature is that with longer sequences, the less influence the older hidden states has on the overall system [54]. This is seen best when looking at Equation 4.3 where with an increase in t, the less effect h_1 would have. This problem is called the 'Vanishing Gradient Problem' and was solved through the development of Long Short Term Memory networks [54].

LSTM

Long Short Term Memory (LSTM), introduced by Hochreiter and Schmidhuber [54], is a type of RNN that overcomes RNN's vanishing gradient problem by explicitly capturing all the long-range dependencies with the input sequence. LSTM's do so through the implementation of a cell state C and three distinct 'gates': the input gate, the forget gate, and the output gate. A gate, σ, refers to the sigmoid functions that control the flow of information found within LSTMs. An illustrative example of an LSTM cell can be seen in Figure 4.10.

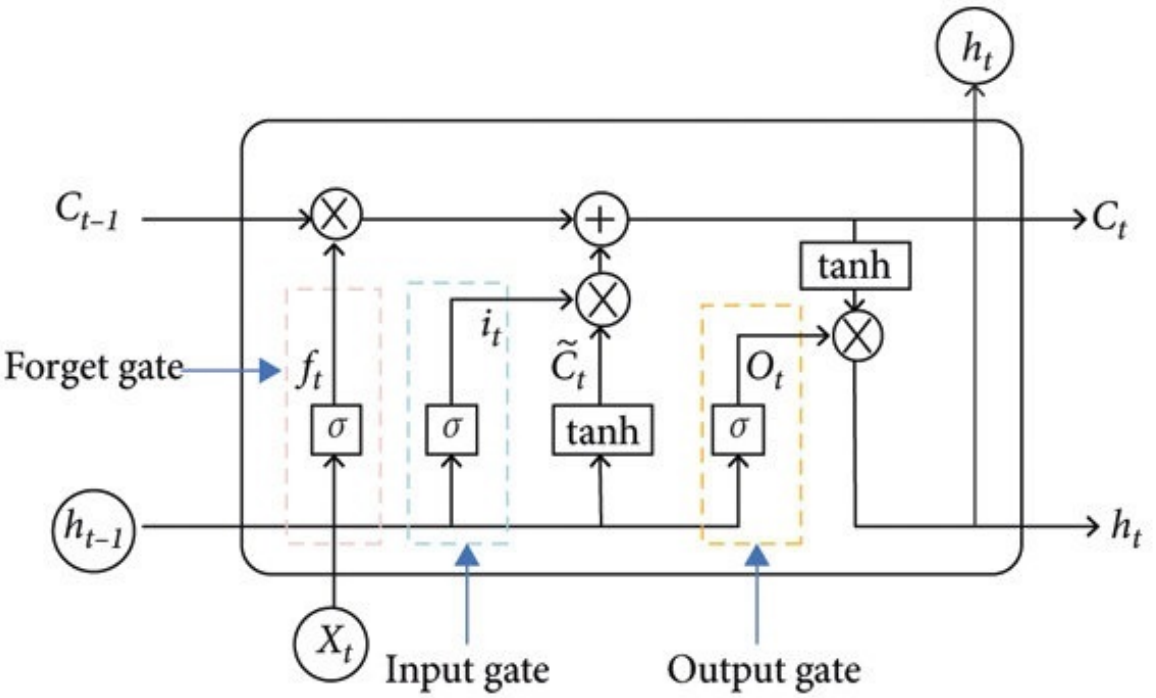

Figure 4.10: Flow of information in an LSTM cell [55].

The flow of information within an LSTM cell is as follows [54]:

- **Forget Gate:**
  This gate takes in the input at time t, x_t, and the hidden state at time $t-1$, h_{t-1}, and passes it through a sigmoid function. This creates the vector f_t which, as representative of the gate used to produce it, represents which information of the cell state at time $t-1$, C_{t-1}, should be forgotten.

$$f_t = \sigma(U_f \cdot x_t + V_f \cdot h_{t-1} + b_f) \tag{4.5}$$

- **Input Gate:**
  Similar to the forget gate, this passes both x_t and h_{t-1} through a sigmoid function to create the vector i_t. Alongside this, both x_t and h_{t-1} are passed through an activation function, in this case, a *tanh* function in order to create vector $\tilde{C}_t$. $\tilde{C}_t$ and i_t work together as the former represents the potential values to update C_{t-1} with and the latter determines which of these potential values will actually be passed on to C_{t-1} to create the new cell state, C_t.

$$i_t = \sigma(U_i \cdot x_t + V_i \cdot h_{t-1} + b_i) \tag{4.6}$$

$$\tilde{C}_t = \phi(U_{\tilde{C}} \cdot x_t + V_{\tilde{C}} \cdot h_{t-1} + b_{\tilde{C}}) \tag{4.7}$$

- **Output Gate:**
  Lastly, the output gate passes both x_t and h_{t-1} through a sigmoid function to create the vector o_t. This vector determines which information from C_t shall be passed on to the new hidden state h_t.

$$o_t = \sigma(U_o \cdot x_t + V_o \cdot h_{t-1} + b_o) \tag{4.8}$$

4.3. State of the Art Deep Learning Findings Relating to Research Topic

As the use of Deep Learning for MTSC gains popularity, research in the realm of applying Deep Learning to human control data has steadily increased. The most prominent of these are those conducted by Versteeg [56], Verkerk [57], and de Jong [58]. Each of these researchers applied Deep Learning to human pilot control data with the objective of classifying certain variables used during the tasks. To be specific, Versteeg classified the dynamics of the controlled element, Verkerk classified the display types, and de Jong classified the skill level of the HO in a single-axis tracking task. As de Jong's research is most relevant to the one discussed in this paper, it shall be the focus of this section.

4.3.1. Data

In conducting his research, de Jong utilised data obtained from an experiment conducted by Pool et al. [59] This experiment was performed in the TU Delft SIMONA Research Simulator using a pitch compensatory display as shown in Figure 4.11 with motion feedback both on and off. The system dynamics of the controlled element was a reduced-order linearized model for the elevator-to-pitch dynamics of a Cessna Citation I expressed as:

$$H_{\theta,\delta_e}(s) = 10.62\frac{s + 0.99}{s\left(s^2 + 2.58s + 7.61\right)} \tag{4.9}$$

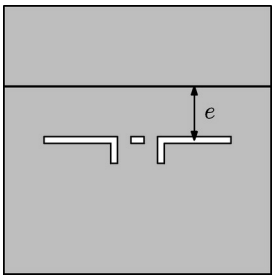

Figure 4.11: Compensatory display used in SIMONA experiment [59].

This data was chosen as it contained time traces of both inexperienced and experienced HO control behaviour as they are found at the start and end of their training phase respectively during the experiment. Additionally, in order to apply the data for training their NN model, they shuffled their data randomly before sectioning 80% of the data for training and the remaining 20% for testing.

Data Labelling

Unlike the research done by both Versteeg [56] and Verkerk [57], there is no clear distinction between the different aspects with which de Jong expects to classify their data. For example, both system dynamics and display type, the focus of Versteeg's and Verkerk's research respectively, are physically different task settings that are chosen by their respective experiment designers. This allows for easier labelling of their data. In contrast, HO skill level is not a physical task setting that can be altered and therefore, de Jong's data is not easily labelled. De Jong, therefore, had to define a method in which they choose to label their data as either skilled or unskilled, for which they considered two possible methods.

The first method was to define skill level based on the experience of the HO which can be seen on the left side of Figure 4.12. In other words, in this method, de Jong would label the data as skilled or unskilled

by their run index defined by parameter L_w, labelling width. This means that the first L_w runs of a HO during their training phase would be considered as 'unskilled' and the last L_w runs of a HO during their training phase would be considered as 'skilled'. This symmetric labelling of data was done to prevent class imbalance and bias in the NN [60, 61].

The second method was to define skill level based on the performance measure such that if the HO's run root mean squared tracking error, RMS(e), was above a certain value, then the run would be considered unskilled and if below another value, then the run would be considered skilled. This labelling method can be seen on the right side of Figure 4.12 and would also be done symmetrically for the same reasons stated in the previous method [60, 61].

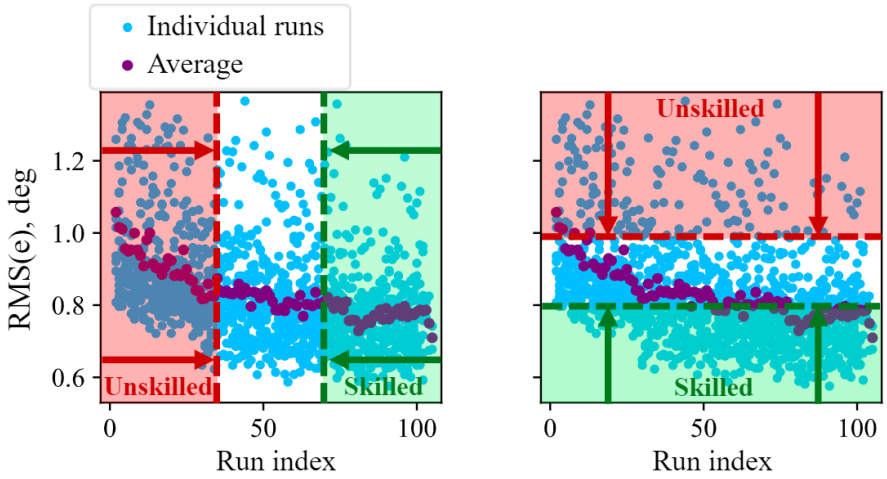

Figure 4.12: Visualisation of both labelling methods proposed by de Jong [58]. The left showcases the experience-based labelling method and the right showcases the performance-based labelling method.

The biggest drawback of both methods as noted by de Jong was the fact that the data was then labelled per run rather than per sample of a run. This preference for the latter was due to the fact that humans are known to be time-invariant that change their control behaviour based on outside factors such as fatigue, loss of attention, or learning [62, 63]. Therefore, this labelling method assumed the fact that humans were time-invariant when it came to skill per run. De Jong however noted that this assumption had to be made as no alternative method existed [2]. Considering the fact that the experiment done by Pool et al. [59] had no sudden changes during their experiment, this assumption was deemed reasonable [2].

After heavy consideration by de Jong, they utilised the first method for their research. This is due to how they define skill level in that a HO's first run in any task cannot be considered 'experienced' or 'skilled' regardless of the differences in performance between participants and vice versa.

Data Pre-Processing

With their data labelled, de Jong then began prepossessing the data in order to optimise the performance of the NN they will use and attain desired classification performance. The methods taken by de Jong were inspired by the methods taken by both Versteeg and Verkerk. In chronological order, the data pre-processing methods that were taken by all three were as follows:

- **Data Scaling:**
  Data Scaling refers to the scaling of data such that the performance of the model converges faster [64] to the highest classification accuracy. In addition, this allows for the standardisation of the data into a common scale such that the NN actively learns the relationship between features with time rather than looking at feature anomalies to perform their classification task. Out of the three researchers, Versteeg was the only one to evaluate different data scaling methods to determine the most optimal one. Versteeg evaluated the following methods:
  - Normalising: Using a maximum absolute scalar to scale the entire sequence between -1 and 1.

$$\hat{u}(t) = \frac{u(t)}{max(|min(\overrightarrow{u})|, |max(\overrightarrow{u})|)} \tag{4.10}$$

- Standardising: Remove the mean μ and scales the entire sequence to the unit variance σ^2, effectively causing the mean to equal zero, $\mu = 0$, and standard deviation to equal 1, $\sigma = 1$.

$$\hat{u}(t) = \frac{\hat{u}(t) - \mu}{\sigma} \tag{4.11}$$

- Robust Scaling: Removes the median and scales the time sequence to its inter-quartile range, IQR.

$$\hat{u}(t) = \frac{u(t) - median(\overrightarrow{u})}{IQR} \tag{4.12}$$

In addition, Versteeg had the choice of applying these methods to either each window or the entire tracking run. The former was beneficial due to the fact that it allowed for real-time usage of the NN in classifying HO behaviour. However, Versteeg noted that when the WS is too small, there may be insufficient data to perform both standardising and robust scaling. Therefore, the following scaling options were tested by Versteeg:

- Sample-wise normalised
- Entire tracking run normalised
- Entire tracking run standardised
- Entire tracking run robustly scaled

After testing these four options, it was found that standardising through the entire tracking run yielded the highest classifier performance [56]. Therefore, this option was used by not only Versteeg, but also by both Verkerk and de Jong as well. Although this indeed resulted in higher performance, this reduced all the classifiers' ability to perform real-time classification. This is for the fact that standardising through the entire tracking run would require the entire tracking run to be completed first before the data pre-processing methods could be performed and then fed into the NN model. However, it still has the potential of providing real-time feedback as the unique μ and σ parameter of a certain HO could be obtained beforehand such that any of their future runs could be standardised per sample using the obtained μ and σ. This however results in the fact that these values need to be obtained first before any real-time classification can be performed. In contrast, a sample-wise method would circumvent this problem entirely and it was noted by Versteeg that although standardising by the entire tracking run yielded the best performance, the option of normalising per window yielded only a 1% decrease in performance [56].

- **Data Partitioning:**
  Data Partitioning refers to the sectioning of the time series data to increase the number of data sets available and time samples within each data set for the training of the NN model chosen such that the performance increases. De Jong did so through three methods:
  - Window Size (WS), measured in seconds, allowed for the sectioning of the entire tracking run to create more samples. For example, if a tracking run was 100 [s] and the window size was set to 10 [s], there would be 10 available data sets for the NN model to train on. This was done for two major reasons. The first major reason was to exploit Deep Learning's unique characteristic in which a NN model shows an increase in performance with an increase in the number of samples to train on with. The second major reason was to allow for the real-time classification of HO control behaviour. If the model was trained on the entire 100 [s] tracking run, then any future application would either require the need for the entire tracking run which does not allow for any real-time feedback to the human controller or the need for the HO would have to wait for 100 [s] before getting any feedback which would not be meaningful to the HO as its a prediction based on their long term behaviour rather their current short term behaviour.
  - Sampling Frequency (SF), measured in hertz, determines the number of samples that is available per second of time series data. For example, if the tracking run was recorded at 100 [Hz], this means that there would be 100 samples per second of tracking run data. By using a sampling frequency of 50 [Hz] on this data, the number of runs recorded would then be halved resulting in 50 samples per run data. As evident from the example, this was primarily used to reduce the number of samples per data set as it has been shown that time sequence data containing an abundant amount of data results in a degradation of performance in NNs [38].

  - Overlap (OL), measured in percentage, is an extension of the use of window size which determines the amount of overlap each 'window' has with its neighbouring 'window'. For example, if an overlap percentage of 50 % is used, then the second half of data found in window n would be the exact same as the first half of data found in window $n + 1$. Similar to the use of window sizes as the overlap is an extension of it, it is used to allow for better performance from the NN models.

The hyperparameters of these three partitioning methods were first optimised through trial and error by Versteeg resulting in a WS of 1.6 [s], SF of 50 [Hz], and an OL of 90 % yielded the highest performance. De Jong then conducted their own optimisation of these hyperparameter values with Versteeg's values as a baseline. They however kept the OL percentage fixed as it was discovered that higher OL percentages always yielded better results [56]. In their research, de Jong discovered that a WS of 1.2 [s] and an SF of 50 [Hz] yielded the best results. This shows that although minimal, different tasks require different hyperparameters in data partitioning methods in order to produce better performance and showcases the variability that exists in data preparation in increasing the performance of the NN.

- **Variable Selection:**
  Variable Selection refers to the selective choosing of variables and their time derivatives from the MTS data to train the NN model with the goal of increasing performance. Although the use of more variables would theoretically allow for the NN model to obtain better performance as it would allow it better understand the relationship each variable has with one another, some variable combinations would result in trivial results. For example, Versteeg noted that in their research regarding the classification of Controller Element dynamics, if they had used forcing function signals, $f_t(t)$, and controlled element output signals, $x(t)$, this would have yielded a model with an accuracy of 100%. This was due to the fact that the NN model would have learned the relation of both signals to determine the correct Controller Element dynamics. This is undesirable as the objective of their research was to determine the Controller Element dynamics solely from the adaptation of the human control behaviour. De Jong [58] however did not face this dilemma as they classified skill level which does not have such trivial relations between any signals, giving them more variability regarding their signals of choice. De Jong noted that the use of all possible variables did not result in better performance and instead led to a decrease in performance and therefore had to determine the different combinations of variables to improve the performance of their NN model. De Jong discovered that a combination of the error signals, e, the input signals, u, and their first-order time derivatives, $\dot{e}$ and $\dot{u}$, yielded the highest accuracies.

4.3.2. Neural Network

With the data obtained, de Jong had to choose which NN model to utilise in his research. They selected four distinct models as candidates for this purpose: LSTM [54], FCN [41], ResNet [41], and Inception Time [42]. LSTM and Inception Time were chosen due to their proven capability when classifying HO control behaviour in tasks similar to de Jong's data as shown by Versteeg [56] and Verkerk [57] respectively. FCN and ResNet were chosen due to their proven performance in time series classification as showcased by Wang et al. [41]

In order to select one of these four models to use in their research, de Jong compared their average validation accuracy to one another after testing them 30 times with 20 epochs each. As seen from the results shown in Figure 4.13, ResNet obtained the highest accuracy and thus was used for the rest of the research.

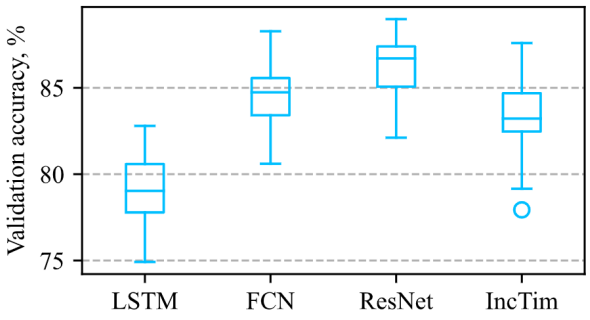

Figure 4.13: Performance of each NN model candidate [58]. Abbreviations: FCN - Fully Convolutional Network, IncTime - Inception Time

The biggest drawback that came from selecting this model was its use of 1-dimensional convolutional layers [58, 41]. Although this allowed the convolutional layer to capture important temporal patterns in the data, as its kernel 'slides' across the data in the temporal dimension, it does not analyse each individual variable and extract their relationships or interactions with one another. This inter-variable relationship is an important aspect of the HO data as their behaviour is directly influenced by the forcing function or the error signal they see. Therefore, not being able to determine these inter-variable relationships with the data presented to it is a limiting factor of the NN model chosen. De Jong theorised that models that utilised both types of convolutional layers either in parallel [43, 45, 46] or in sequence [44] may preserve both the spatial and temporal dynamics found in HO control data.

4.3.3. Results

Once the data has been labelled and pre-processed, de Jong separated the data between runs that had motion feedback and those without. The data was then used to train and validate the model. On average, de Jong obtained a validation accuracy of **87.95%** for data with no motion feedback and **92.14%** for data with motion feedback. This difference in performance signified that, in terms of skill level, motion feedback allowed for more consistently distinguishable characteristics in terms of skill level.

Although the validation accuracy obtained by de Jong showed promising results, they investigated their classifier's performance more extensively, specifically its performance on data of participants it has never seen. De Jong selectively removed the entirety of a single participant's data from both the training and validation data set and used it as a testing data set. This meant that for the entirety of their model's training phase, they were never shown the excluded participant's data set. The rationale behind this was the intended real-life use of the classifier. As the human population is simply too large to obtain tracking data for every single person, this implies that the classifier will have to classify tracking data either from HOs it was never trained on. Therefore, a performance indicator of the classifier in such a situation would be significant in its applicability in the real world.

De Jong's test of removing one participant's data for the training phase of the model was done for each participant with the average performance across all excluded participants being the model's quantitative measure of performance in such situations. They obtained an average test accuracy of **62.36%** for data with no motion feedback and **75.13%** for data with motion feedback. The obtained test accuracy is shown to be significantly lower than the obtained validation accuracy.

To help visualise why there was a sudden drop in accuracy, de Jong plotted the prediction of skill with an increasing number of runs for every participant when they were removed from the training and validation dataset for the runs with no motion feedback. This figure is shown below in Figure 4.14. In this figure, every marker indicates a subject's individual run and the color indicates the average model output for each run. Starting with the top graph, it can be seen that even though an experience-based labelling method was used, the model essentially based their classification skill level on the error signal as the transition of unskilled to skilled is from top to bottom rather than from left to right.

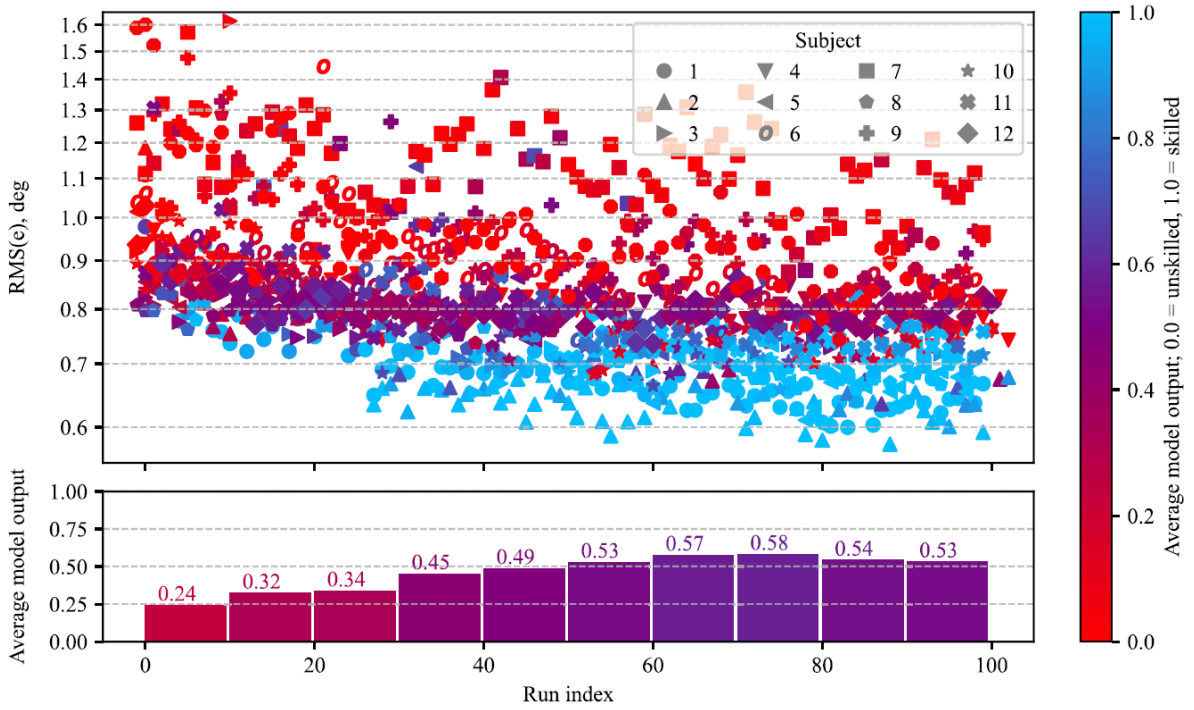

Figure 4.14: Each marker in the top figure represents the individual tracking run for an individual subject with the color indicating the average model out of the respective run. The bottom figure presents the average model output for bins of run indices. These figures were plotted with data from runs with no motion feedback [58].

This process was also done for the run with motion feedback with the plot shown in Figure 4.15. In contrast to Figure 4.14, the figure shows a more desired output where the transition from unskilled to skilled is from left to right rather than from top to bottom. This suggests the presence of two phenomena. The first is that the presence of motion feedback allows for more discernible behaviour between performances at the beginning of training and at the end of training. The second is the fact that in order for the model to perform as expected, it utilises more than only the error signal in making its classification.

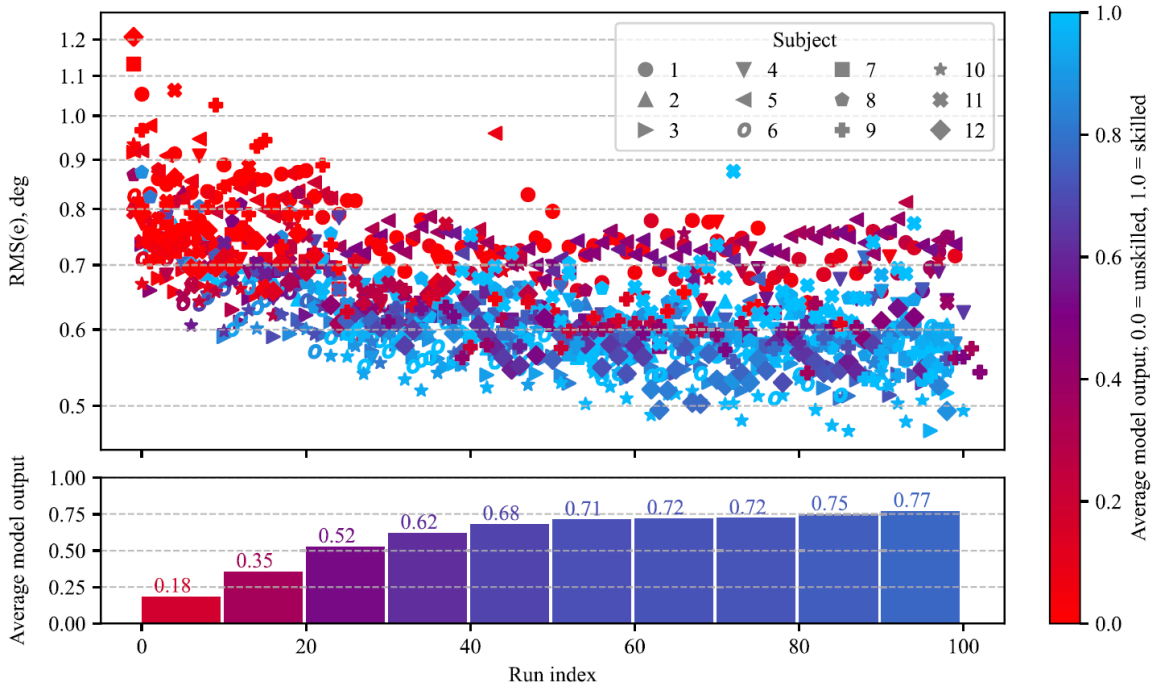

Figure 4.15: Each marker in the top figure represents the individual tracking run for an individual subject with the color indicating the average model out of the respective run. The bottom figure presents the average model output for bins of run indices. These figures were plotted with data from runs with motion feedback [58].

The overall drop in accuracy from validation to test datasets was theorised by de Jong to be caused by the labelling method of the data. This was due to the non-linear behaviour of human skill levels. For example, assuming that the HOs were to be rated out of 10 based on their control performance, a HO may begin training with a rating of 4 and end training with a rating of 6 whereas another HO may begin with a rating of 6 and end with a rating of 8 yet both the performance that resulted in a 6 would be classified differently. This data 'mislabeling' was hypothesised to have caused a reduction in accuracy during both training and testing. De Jong however noted that this was the complexity that came with classifying human skill level, a non-linear characteristic of humans. De Jong, therefore, stated that increasing the amount of HO data would be the best method in order to improve the NN accuracy. As they were unable to obtain more HO data and the fact that obtaining HO data for every single person would be impossible, de Jong theorised that synthesising HO control data through the use of Cybernetics would be able to overcome this limitation regardless of the fact that it is unable to simulate the non-linear behaviour of humans in the form of the remnant. However, it was shown that using such synthesised data led to a decrease in performance when using said data solely for training the classifier as it was theorised the missing remnant was of great importance in the classification of skill level in HOs and thus does not allow for effective training-data augmentation [65].

4.3.4. Explainability

De Jong also delved into the possibility of adding explainability to their classifier to help explain the logic that was taken by the model in making their prediction regarding skill level. By utilising a method called SHAP [66], a concept which will be explained further in Subsection 3.5, they were able to deduce which input variable was found to be most important when the model makes a classification. Though SHAP determines the importance of the variables to a singular input, a local explanation, it is possible to combine a very large set of local explanations in order to create one that explains the overall working of the model, a global explanation. By doing so, de Jong wa able to obtain this global explanation for his model for the no-motion dataset as shown in Figure 4.16. As noted by de Jong, the error signal e had the highest contribution when determining whether a tracking data was skilled or unskilled with the input signal u having the lower contribution.

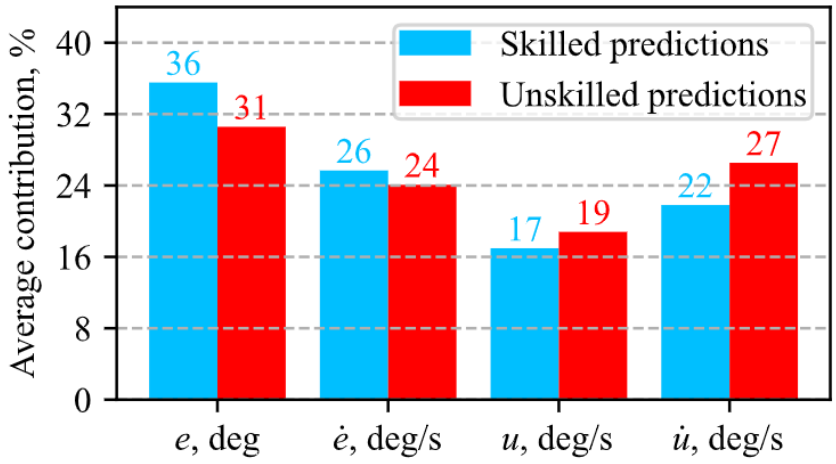

Figure 4.16: Global feature importance expressed as an average contribution to model classification for samples predicted as 'skilled' or 'unskilled' [58].

Alongside the use of SHAP, de Jong [58] utilised a technique called 'Activation Maximisation' [67] which helps create a visualisation of what variables the neuron in the CNN is detecting in order to make a classification. In other words, it creates input signals that maximally activate the neurons. The results obtained from this technique is shown in Figure 4.17. As was noted by de Jong, the class visualisation of the 'skilled' sample is notably much smoother than that of the 'unskilled' sample which appears to be more variable and noisy. De Jong attributed this to the presence of stronger remnant found within unskilled pilot control behaviour.

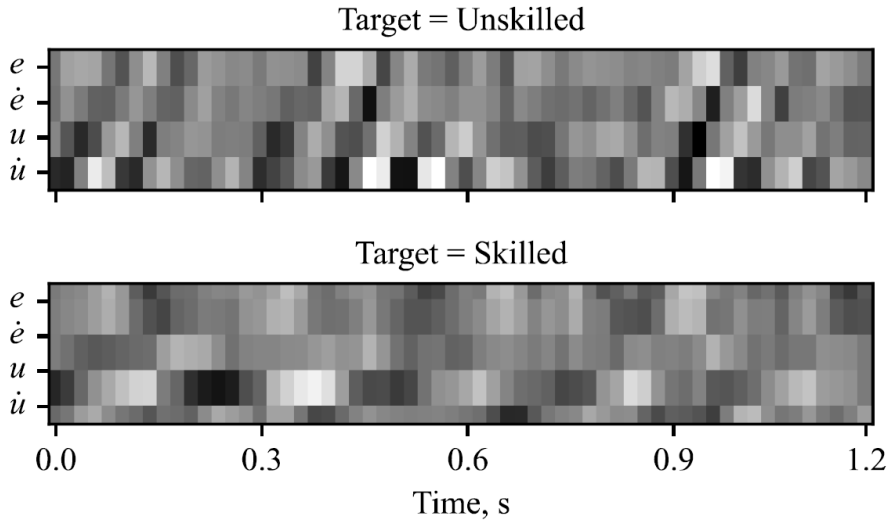

Figure 4.17: Class visualisations generated using Activation Maximisation [58].

Overall, when taking into account the results obtained through the use of SHAP and Activation Maximisation, de Jong determined that when making 'skilled' predictions, the model would put more emphasis on the error signal and its derivative whereas when making 'unskilled' predictions, it would put more emphasis on the input signal and its derivative. De Jong speculates that the use of the input signal and its derivative in making 'unskilled' predictions by the model is due to the model's ability to detect a significantly greater amount of non-linear behaviour, or remnant, in these signals between 'unskilled' and 'skilled' HO. De Jong speculates this as it was found that non-linear behaviour in these signals is much higher in

untrained pilots than in trained pilots [30]. However, they acknowledged the fact that the use of SHAP came with its drawbacks as it requires a simplified explanation model to interpret the actual NN model and therefore has the possibility to be inaccurate in its presentation while also being computationally expensive. The same can be said with Activation Maximisation as although it generates an input that activates the neurons when making a prediction of a certain class, it does not explain any interpretable insights into the network's decision-making process. De Jong recommended exploring the possibility of using other forms of explainability methods that explain the actual NN model itself by delving into the model's training parameters would present a more accurate explanation of the logic behind the classifier.

4.4. Selecting Neural Network Models for Multivariate Time Series Classification

This section presents the criteria used to obtain the most appropriate NN models for this research along with a list of said models.

4.4.1. Criteria in Selecting Models

With the number of deep learning architectures currently being developed or already been developed for Multivariate Time Series, it was essential to determine which models to use in this research. The motivation behind choosing the models is not solely determining which one has shown to yield the highest accuracy because the goal of this research is to determine the feasibility of using deep learning in classifying skill levels in dual-axis tracking tasks. Therefore, the main factors that determined the models that would be used in this research are as follows along with their motivation:

- **Performance:** As the goal of this research is to determine the feasibility of using deep learning to assess the HO's skill level, it is essential to use a model which has the potential to effectively do so. This is done by looking at the model's average accuracy when ran with various MTS data.
- **Implementation:** Due to time constraints, it was necessary to select models which would allow for swift implementation. This was done by determining whether there were already existing implementations of the model in the programming language and framework chosen for this research.
- **Complexity:** Another aspect that needed to be addressed due to time constraints is the complexity of the models chosen, specifically the number of parameters or weights the model has. This is because the greater number of parameters the model has, the longer the training phase would take for a model and the more computing memory it needs [50]. Ideally, a model with a lower number of parameters would be chosen though this could potentially lead to lower performance. Therefore, a balance would need to be achieved between performance and complexity.

4.4.2. Chosen Neural Network Models for Multivariate Time Series Classification

After going through numerous NN models with the criteria in mind, five models were determined to be most viable for this research which can be grouped as follows:

- **Temporal Analysing Models:** NN models that analyse information relative only to the time dimension.
  - ResNet [41, 68]
  - MLSTM-FCN [43]
- **Variable and Temporal Analysing Models:** NN models that analyse information relative to both the variable and time dimension.
  - MTEX-CNN [44]
  - XCM [45]
  - TSEM [46]

Models that were considered but were not chosen for further use in this research as they did not meet the aforementioned criteria are as follows: TapNet [69], Disjoint-CNN [70], ShapeNet [71], TCRAN [72], and DA-Net [73]. Note that ResNet was not assessed under the same criteria as the other four chosen models and was added to act as a link between this research and the research done by de Jong in using ResNet to assess a HO's skill level in single-axis tracking tasks. Furthermore, when implementing these models, the hyperparameters used were the ones that were explicitly stated in their respective papers. If none were found, the hyperparameters of the other models were applied to the model.

ResNet

Residual Networks (ResNet) was introduced by He et al. [68] as a framework to train very deep NNs. As explained by Simoyan and Zisserman [74], with an increasing amount of convolutional layers, or simply, the deeper the Convolutional NN, the better the performance of said NN. As a result, Wang et al. [41] propose the use of residual blocks to develop the model shown in Figure 4.18 which consists of three residual blocks with three layers each, effectively creating a model with nine convolutional layers. In addition to this, He et al. introduced the use of residual connections which act as 'shortcuts' between the layers to enable gradient flow directly through the bottom layers of each residual block. These connections were also adopted by Wang et al. as seen in Figure 4.18. The final model produced by Wang et al. [41] was also used in de Jong's research [58].

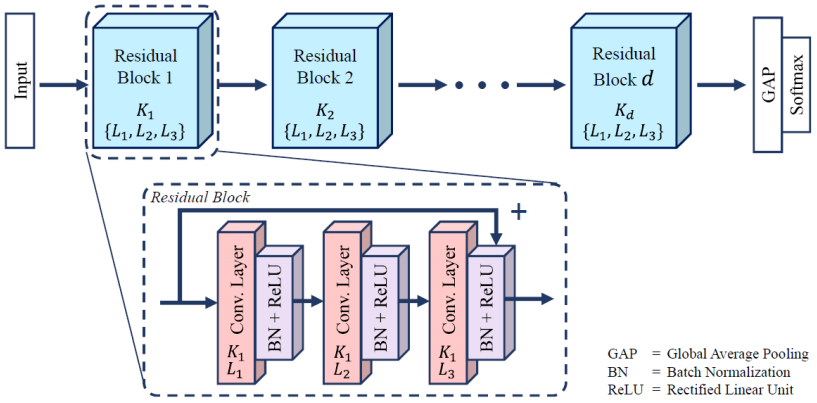

Figure 4.18: Network structure of ResNet [58].

MLSTM-FCN

Multivariate Long Short Term Memory - Fully Convolution Networks (MLSTM-FCN) and Multivariate Attention Long Short Term Memory - Fully Convolution Networks (MALSTM-FCN) were introduced by Karim et al. [43] as a multivariate modification of the univariate time series classification models Long Short Term Memory - Fully Convolutional Network (LSTM-FCN) and Attention LSTM-FCN (ALSTM-FCN) by Karim et al [75]. They do so by introducing a squeeze-and-excitation block that adaptively recalibrates the feature maps produced by the convolutional layers in the model. This block is the biggest factor in the improved accuracy for MTSC when compared to its univariate version as it allows for the possibility for the network to learn the inter-correlations between the multiple dimensions or variables at each time step, a feature that was not present at LSTM-FCN and ALSTM-FCN [43]. Although the paper shows an increase in performance when using MALSTM-FCN compared to MLSTM-FCN, the latter will be used for the rest of this research due to its already available implementation in the framework of choice, `PyTorch`. An illustration of the network structure can be seen in Figure 4.19.

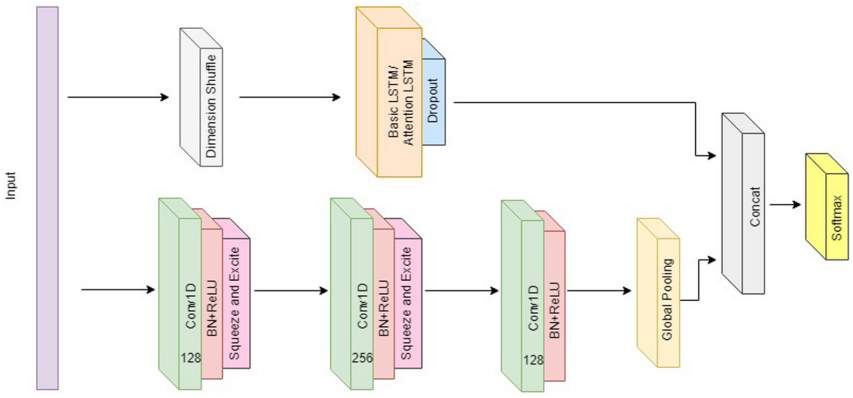

Figure 4.19: Network structure of MLSTM-FCN / MALSTM-FCN [43].

MTEX-CNN
Multivariate Time Series Explanations for Predictions with Convolutional Neural Networks (MTEX-CNN) was introduced by Assaf et al. [44] as a novel explainable CNN model which would allow for both the making of predication based on MTS data and explain said predictions through the use of Grad-CAM. The overall network structure is illustrated in Figure 4.20. MTEX-CNN is split into two stages with each stage analysing different parts of the data to determine which are most significant in making a prediction. Specifically, the first stage, highlighted in green, aims to explain which variables are most significant in making a prediction and the second stage, highlighted in red, aims to explain which time segments are most significant.

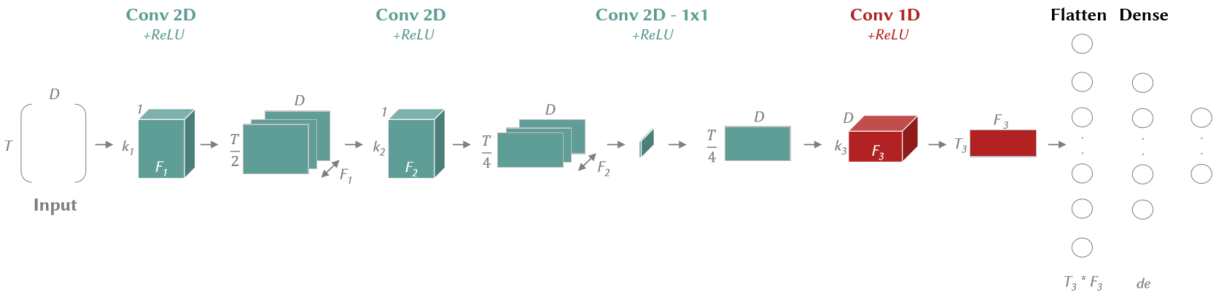

Figure 4.20: Network structure of MTEX-CNN [45].

XCM
Explainable Convolutional Neural Network for Multivariate Time Series Classification (XCM) was introduced by Fauvel et al. [45] as a compact convolutional NN which extracts information relative to the observed variables and time directly from the input MTS data. Similar to the models MLSTM-FCN and MALSTM-FCN, the model consists of two channels as shown by Figure 4.21 which illustrates the network's overall structure. The upper channel, highlighted in green, extracts information relative to the observed variables in the MTS data with the lower channel, highlighted in red, extracts information relative to time. The output feature maps are then concatenated and passed through to one last convolutional layer to allow for the analysis of the relations between the variables and the time segments.

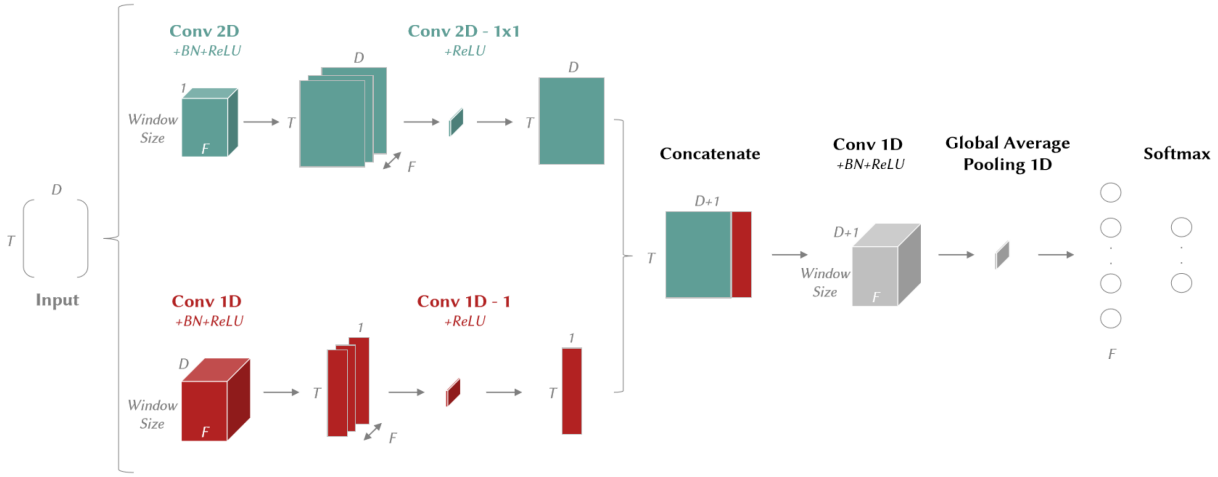

Figure 4.21: Network structure of XCM [45].

TSEM
Temporally-Weighted Spatiotemporal Explainable Neural Network for Multivariate Time Series (TSEM) was introduced by Pham et al. [46] as a novel NN method that, similar to MLSTM-FCN and MALSTM-FCN, combines both RNNs and CNNs. TSEM's network structure is illustrated in Figure 4.22 and it can be

seen that it is extremely similar to the network structure of XCM, illustrated in Figure 4.21. The biggest difference between the two is TSEM's replacement of XCM's lower channel's convolutional layers with an LSTM layer alongside the multiplication of the generated feature maps from each channel rather than a concatenation of the two. Overall, TSEM uses the extracted global temporal features from the RNN channel, as highlighted in purple, and uses it as an importance weight vector for spatiotemporal feature maps generated by the CNN channel, as highlighted in red. Similar to XCM, this resulting feature map is then passed through a convolutional layer to allow for the analysis of the relations between the variables and the time segments.

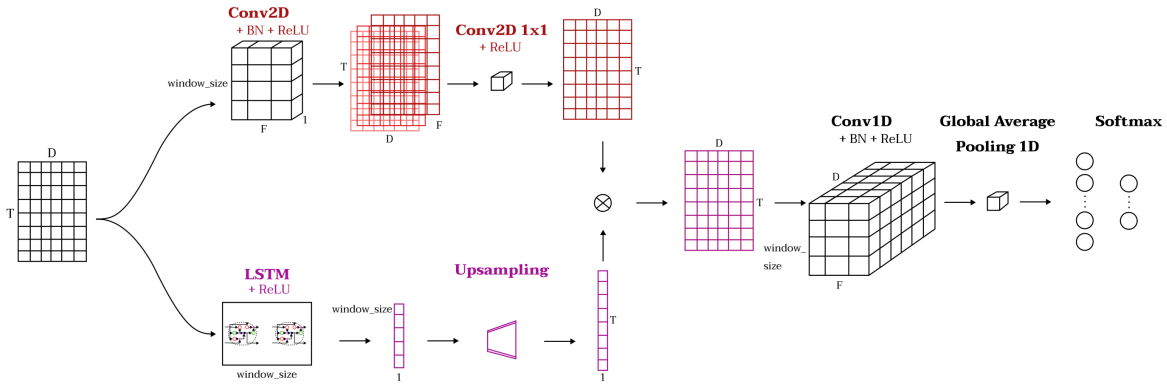

Figure 4.22: Network structure of TSEM [46].

4.5. Explainable Artificial Intelligence

Deep learning is infamous for its black-box behaviour, a system whose inputs and outputs are clearly defined but whose inner workings are incomprehensible [76]. Put simply, when a NN network is trained for a task, it is difficult to understand both how the neurons work together to make a decision and what each particular neuron is doing on its own. Although it is possible to extract the weights and biases of each neuron to understand the processes taken by the NN, most NN contain thousands of neurons and therefore makes the analysis of each weight and bias inconceivable. This lack of transparency regarding the logic behind NNs is one of the biggest drawbacks of Deep Learning [77]. To overcome this, Explainable Artificial Intelligence (XAI) as a topic has seen an increase in popularity to understand better the decision-making steps that are taken by these NNs. XAI within the realms of Deep Learning are post-hoc in nature and can be separated into two distinct types [78]:

- Model-Specific Explainability
- Model-Agnostic Explainability

4.5.1. Model-Specific Explainability

Model-Specific Explainability refers to XAI methods which exploit the special characteristics of certain types of NNs, hence the model-specific nomenclature. These methods can only be applied to the NN types they were designed for.

Model-Specific Explainability in CNNs

The most prominent form of Model-Specific XAI methods is Class Activation Mapping (CAM) introduced by Zhou et al. [79] which is only applicable to NNs that contain convolutional layers and requires the presence of a Global Average Pooling (GAP) layer after the final convolutional layer. This requirement is because the GAP layer essentially takes in the feature maps produced by the preceding convolutional layer and 'squeezes' each one into a singular scalar value which is then fed into an FC layer with an activation function to predict a class assignment. The weights of the FC layer are then used to determine the weighted average of the produced feature maps with a higher one representing a more important feature map and vice-versa. This process is illustrated best in Figure 4.23. Although CAM was made with image classification tasks in mind, CAM is also applicable to TSC tasks as shown in the right figure of Figure 4.24.

Figure 4.23: Producing the CAM of a NN classifying an image by obtaining the weighted average of the feature maps [79].

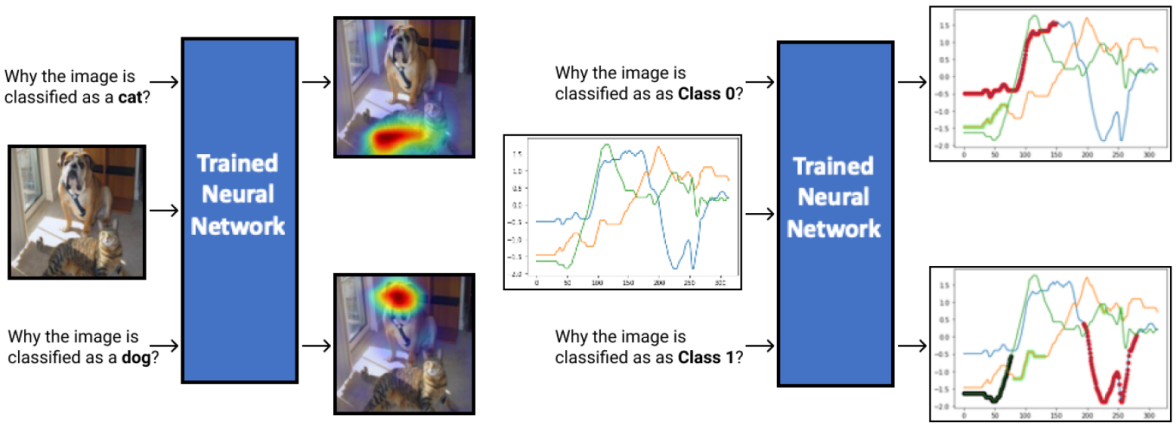

Figure 4.24: Explanations regarding the NN's reasoning behind its classification given by CAM [46].

The major drawback that arises from CAM is its lack of versatility which stems from the mandatory requirement of a GAP layer. This, therefore, limits the possible use of CAM on models that, either by coincidence or choice, have this implemented. To circumvent this, different CAM methods have been developed which can be separated into two distinct categories. These categories are shown below along with a few examples of the different methods of each category. Note that although CAM is listed, this is simply to illustrate the category of CAM methods it belongs to and therefore still requires the presence of a GAP layer.

- **Activation-Based Methods:** CAM methods that generate a class activation map through the use of the feature maps of the last convolutional layer and the feature map's activation scores of the final fully connected layer.
  - CAM [79]: The first instance of an Activation-Based CAM
  - Score-CAM [80]: An improvement of CAM as it does not require the model to contain a GAP layer while also yielding tighter accuracy and precision [80]
- **Gradient-Based Methods:** CAM methods that generate a class activation map through the use of the gradients of the output class score with respect to the feature maps. These methods do not have

the limiting requirement of a presence of a GAP layer. The use of gradients results in the methods requiring a higher computational workload.

- Grad-CAM [81]: The first instance of a Gradient-Based CAM. Utilises first-order gradients to obtain its class activation maps [79].
- Grad-CAM++ [82]: An improvement of Grad-CAM in terms of accuracy and precision however results in worse class-discriminative abilities [83]. Similar to Grad-CAM but utilises second-order gradients to obtain its class activation maps [82].
- XGrad-CAM [83]: An improvement of Grad-CAM in terms of accuracy and precision and class-discriminative abilities [83]. Similar to Grad-CAM in utilising first-order gradients to obtain its class activation maps [83].

Because all the selected NN models contain convolutional layers, this XAI method is suitable for this research and will be used. However, only Grad-CAM, Grad-CAM++, and XGrad-CAM will be utilised in this research as it was found that compared to other Activation-Based and Gradient-Based CAM methods, they produced the most accurate and precise class activation maps [46].

Model-Specific Explainability in RNNs

Model-Specific Explainability in RNNs is not as abundant as those for CNNs. This was shown when Arrieta et al. [78] found only 10 papers regarding XAI with RNNs compared to the 32 papers regarding XAI with CNNs. Although methods exist to allow for post-hoc explainability in RNNs as shown by Samek et al. [84] and Karpathy et al. [85], they are applicable only to natural language processing. Although there is an RNN-exclusive XAI method that allows for the visualisation and understanding of RNNs called the 'Attention' mechanism [86], this mechanism is not a post-hoc XAI method and is required to be actively built in by the model architect. As none of the models used in this research contains an Attention mechanism, this method will not be used in the research.

4.5.2. Model-Agnostic Explainability

Model-Agnostic Explainability refers to XAI methods that are designed to be utilised with NN models regardless of their network structure. These methods are therefore the most versatile.

LIME

Local Interpretable Model-Agnostic Explanations (LIME) is an XAI method introduced by Ribeiro et al. [87] that explains the predictions made by a network by creating a simpler, interpretable model that is trained locally on the prediction of the original model on that instance. This, therefore, means that LIME can only provide an explanation that is locally faithful and does not present a global explanation. However, because LIME was not implemented for the deep learning framework of choice, PyTorch, it will not be used in this research.

SHAP

Shapley Additive Explanations (SHAP) is an XAI method introduced by Lundberg and Lee [66] that explains the predictions made by a network by using game theory to obtain Shapley values, put simply the importance value, of each input variable. The most significant difference between the SHAP and LIME is SHAP's ability to showcase both local explainability and global explainability by aggregating the local Shapley values of each input. This means that SHAP is able to explain which variables of the entire dataset were most important when making predictions. However, due to the implementation of SHAP being more difficult than CAM, it shall not be featured in this research.

4.6. Key Takeaways

This chapter explained background information on the current standards of NNs regarding MTSC. Specifically, it introduced the inner workings behind a simple FCN alongside an exploration of the two most common NN archetypes used in TSC, CNNs and RNNS. Afterwards, a review of modern literature regarding the use of NNs in both classifying skill levels based on HO behaviour and MTSC and its explainability. The key takeaways of this chapter are as follows:

- As discovered by de Jong, utilising NNs in assessing HO skill shows great promise in classifying the skill level of human control behaviour in single axis tracking tasks with an obtained validation

accuracy of 87.95% for data with no motion feedback and 92.14% for data with motion feedback. The biggest drawback however came with its obtained testing performance which attempted to simulate the model's performance in real life. The model obtained an average test accuracy of 62.36% for data with no motion feedback and 75.13% for data with motion feedback. The most probable causes for this were first, the use of only 1-dimensional or 2-dimensional convolutional layers rather than a combination of both and second, the low number of participants available for training the network. Overcoming these causes could theoretically result in an increase in performance from a model in classifying skill levels based on HO control behaviour.

- As shown by the favourable validation performance of the model produced by de Jong, the data preprocessing methodology they utilised was compatible with HO control behaviour data and will therefore be utilised for this research. However, the variables of the HO control behaviour data to be used as inputs for the NN model has to be optimised as it was shown in Figure 4.14 and Figure 4.15 that for the model to perform as expected, the model has to utilise signals other than the error signal.
- After reviewing multiple NN models designed for MTSC under three defined criteria, performance, implementation, and complexity, four models were chosen to be used for the rest of the research alongside the model used in de Jong's research, ResNet [58]. They are:
  - ResNet
  - MLSTM-FCN
  - MTEX-CNN
  - XCM
  - TSEM
- Different forms of XAI methods applicable to NNs were explored to add an explainability aspect to the model that will be developed. After looking at both model-specific and model-agnostic methods of deep learning XAI methods, it was deemed that CAM [79], a model-specific XAI method would be used for the rest of the research due to its ease of implementation and clarity in explanation.

5

Research Objective and Questions

In Chapter 3, the limitations found within the cybernetic approach in assessing human operator skills levels along with the state of the art in Cybernetics research in multi-axis tracking tasks were introduced. In Chapter 4, a summary of state-of-the-art applications of NNs in MTSC is introduced and showcases how it would be able to overcome the limitations found within the cybernetics approach. Therefore, the following research objective can be formulated:

> *To investigate the effectiveness of state-of-the-art Deep Learning Methods in recognising and properly classifying the skill level of a human operator's applied control strategy in a two-axis tracking task whilst providing transparency regarding the decisions that were made to create the classification.*

In order to support the accomplishment of the research objective, the following main research question is formulated:

> *To what extent can Deep Learning be used to classify a pilot's skill level in a two-axis control task?*

As this main research question is too complex to solve in and of itself, it shall be split into research sub-questions to clarify the scope of the research and focus on specific aspects of the research objective. These sub-questions apply to the entirety of the research project, not solely to this literature study, and are as follows:

1. How can Deep Learning be used to classify human behaviour in dual-axis tracking tasks based on skill?
   (a) Which of the proposed NN models could produce meaningful results?
   (b) What NN model-specific hyperparameter values would provide the highest level of accuracy?
   (c) Which combination of signals and time derivatives would yield the highest level of accuracy?
2. To what extent can the robustness of the classifier be determined?
   (a) What is the optimal performance obtained by the classifier?
   (b) How well does the classifier perform on never before seen subject tracking data?
   (c) How well does the classifier perform on single-axis tracking data?
3. How can an explainability aspect be added to the classifier?
   (a) How can CAM be used to interpret the classifier's decision-making?
   (b) What input parameters are most important in classifying the skill level of tracking data?

6

Preliminary Analysis

This chapter aims to report on the results obtained from a preliminary analysis based on the research topic. This preliminary analysis was performed for two purposes: to act as a proof of concept and to answer **Sub-question 1(a)** which was introduced in Chapter 5. Section 5.1 begins with where the data that will be used for the preliminary and main phases of this research originated from and how it was pre-processed for the NN model. The following section, Section 5.2, then presents how the NN models were implemented. Section 5.3 then presents the results with Section 5.4 showing to what extent CAM can be used to extract information from the two best NN models as evaluated from the previous section. Lastly, Section 5.5 presents the logic, results, and reasoning behind the creation of a new NN model that will be used for the rest of the research.

6.1. Data Handling

6.1.1. Data Origin

The data used in the preliminary test originated from the experiment performed by Wijlens et al. [30]. In their paper, Wijlens et al. objectively and quantitatively evaluated the acquisition, decay, and retention of skill-based manual control behaviour in a compensatory dual-axis roll and pitch attitude tracking task. The data obtained in the paper is compatible with the current research at hand due to the presence of training runs within the experiment in which participants undergo training for four days with 25 runs per day. These training runs will be used for the rest of the research.

In the experiment, as stated before, participants were tasked to perform a compensatory dual-axis pitch and roll attitude tracking task. In order to make the task as realistic as possible, each axis had its own distinct dynamics. The linearised dynamics are shown below with pitch dynamics represented by Equation 6.1 and roll dynamics represented by Equation 6.2.

$$H_{c\theta}(s) = \frac{\theta}{\delta_e} = \frac{0.33282\left(s^2 + 0.09244s + 0.002886\right)}{\left(s^2 - 0.01388s + 0.004072\right)\left(s^2 + 0.446s + 0.4751\right)} \tag{6.1}$$

$$H_{c\phi}(s) = \frac{\phi}{\delta_a} = \frac{0.76773\left(s^2 + 0.2195s + 0.5931\right)}{(s + 0.7363)(s - 0.01984)\left(s^2 + 0.1455s + 0.6602\right)} \tag{6.2}$$

These dynamics were chosen as they are the controlled aircraft dynamics of a medium-sized twin-engine transport aircraft that is similar to a Boeing 757 in terms of size [30], thus ensuring that the task is as similar to the actual piloting of an aircraft as possible. Due to the need for the operator to adopt lead equalization by both dynamics, the task is rather challenging to perform [5, 7]. This allows for the possibility of more consistent and distinguishable discrepancies when comparing a participant's tracking data during their initial unskilled runs and their final skilled runs. In contrast, an easier task, such as one in which the task's dynamics require no form of either lead or lag equalization, would likely not produce such consistent and distinguishable discrepancies.

The experiment was performed in the fixed-base simulator setup, therefore with no motion feedback, in the HMILab at TU Delft. The display used is shown in Figure 6.1 with e_θ representing the pitch error and e_ϕ representing the roll error.

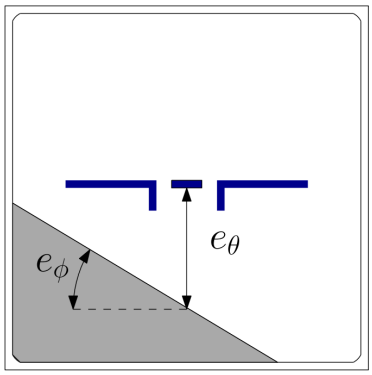

Figure 6.1: Dual-axis compensatory visual display [88].

6.1.2. Data Preparation

Similar to what was done in the research done by de Jong [58], the data obtained needed to be prepared to allow for classification purposes by the NNs. The data will be prepared using the same steps that were done by de Jong [58] as was discussed in Section 4.3.1. However, as this is a preliminary test, no optimisation regarding the steps will be done. Therefore, the values and methods chosen by de Jong will be utilised here. These values and methods are shown in Table 6.1. In addition, to reduce the computational workload, only runs from 12 subjects were used instead of the complete 38 subjects.

Table 6.1: de Jongs Optimal Pre-Processing Hyperparameters and Methods [58]

Hyperparameter	de Jong's Optimal Values or Methods
Label Width [runs]	20
Sampling Frequency [Hz]	50
Window Size [s]	1.2
Overlap Percentage [%]	90
Input Signals [-]	$e + \dot{e} + u + \dot{u}$
Scaling Methods [-]	Standardizing Entire Tracking Run

As this current research involves dual-axis tracking tasks, the input signals used for classification purposes will be signals used by de Jong for both axes, i.e. $e_\theta + \dot{e}_\theta + e_\phi + \dot{e}_\phi + u_\theta + \dot{u}_\theta + u_\phi + \dot{u}_\phi$.

6.2. Neural Network Implementation

For the preliminary experiment, all the models will be trained using the prepared data as discussed in the previous chapter. This is done to determine which of the models to use for the main experimental phase of the research. The NN models were programmed in `Python 3.7` using `PyTorch v1.13.1` [89]. The computer used to perform this preliminary experiment is equipped with an NVidia P100 Graphical Processing Unit (GPU).

6.2.1. Neural Network Hyperparameters

As noted in Chapter 4, NNs contain settings known as hyperparameters which can be tuned to allow for better performance. Because this is currently a preliminary experiment, the model-specific hyperparameters chosen will be those used in the corresponding research papers. One model-specific hyperparameter that was not set to a singular value was the 'window size' hyperparameter found in both XCM [45] and TSEM [46]. This hyperparameter determines the kernel size and hidden cell size which the convolutional layer and LSTM cell use respectively and is a percentage of the length of the entire time sequence. For example, if the MTS has a length of 60 units with a window size of 50% applied, then the kernel and hidden cell size are 30 units. Within the XCM paper, they tested a range of window sizes for each dataset to determine the most optimal value for each. As this is a preliminary experiment, this shall not be done and instead, a value of 80% window size shall be used as it was the value that yielded the highest accuracies

for the most amount of datasets [45]. Note that this window size is separate from the one used to partition the dataset during the data preparation stage. Another model-specific hyperparameter that will need to be adjusted is the number of feature maps produced by the convolutional layers of each model. This hyperparameter will be adjusted such that the highest number of convolutional filters is equal to 256. This number was chosen as it was the highest number of convolutional filters seen throughout all the models and was therefore set as the standard. Note that for models which have various convolutional layers, the layer with the highest number of filters will be increased to 256 with the other layers increased such that the ratio stays constant. For example, with the MTEX-CNN model [44], the highest filter amount was 128 with the lowest filter amount being 64, resulting in a filter ratio of 2:1. Therefore, for this preliminary test, the highest filter amount was increased to 256 with the layer with 64 filters increased to 128 such that the ratio remains 2:1.

The values of model agnostic hyperparameters, batch size, epoch, loss function, optimiser function, learning rate, and output activation function, are shown in Table 6.2. The value for batch size was chosen to allow for a decrease in training time with only a slight cost of training stability and generalisation performance [90]. The epoch value was chosen to allow for convergence regarding performance as a lower value would not allow for the model to properly converge in terms of performance whereas a higher value would result in longer training times with diminishing return in a performance increase. The loss function chosen was determined based on the fact that out of all the papers that explicitly stated their loss function, they all used the same loss function. This methodology was done for the optimiser function, learning rate, and output activation function. Each model was trained only once to avoid multiple long-running training runs.

Table 6.2: Model Agnostic Hyperparameters

Hyperparameter	Value
Batch Size [-]	128
Epoch [-]	30
Loss Function [-]	Categorical Cross Entropy
Optimiser Function [-]	Adam [91]
Learning Rate [-]	$1e-3$
Output Activation Function [-]	Softmax

6.3. Results

The validation accuracy for each of the models after training for 30 epochs is shown in the top left of Figure 6.2. As can be seen in this figure, both ResNet and MLSTM-FCN achieve extremely high validation accuracies of above 98% with XCM coming close with a score of 93.63%. TSEM and MTEX-CNN were the models with the lowest validation accuracies, achieving a score of around 90%. If these results would be used to make a decision as to which NN models are to be used for the next stage of this research, ResNet and MLSTM-FCN would be utilised. However, this shall not be done. This is because the purpose of the NN model being built is to excel not only on the data that it was trained on, but also on the general population, i.e. tracking data from human participants the model has never been trained on. Therefore, a higher validation accuracy could mean that the model learns 'too well' on training data and picks up 'noise' in each participant's data rather than the underlying patterns. Therefore, the models were tested on an aptly named test dataset, a dataset of a participant with which the models never trained on. In this case, participant 4's data was used as the test dataset. The obtained accuracy is then named the testing accuracy with which the model's results are presented in the top right of Figure 6.2.

As can be seen in the resulting figure, all the models show a significant drop in performance compared to their validation accuracy. The drop in performance in the models is calculated and shown in the bottom of Figure 6.2. The most noticeable of this performance dip is in the TSEM model with a drop of more than 39%. This essentially lead to the model having a test accuracy of around 50%, which meant the model was essentially labelling all the tracking traces at the same skill level. The second largest drop in performance came from the XCM model with a drop of 26.75%. Both the ResNet and MLSTM-FCN model were the top performers regarding testing accuracy with a similar drop in accuracies, around 17%. Remarkably, MTEX-CNN showed the lowest drop in accuracies across all models with a drop of 11.28%

whilst maintaining relatively high validation and testing accuracies. This shows MTEX-CNN's ability to understand and make classification predictions based on underlying patterns rather than participant-specific noise.

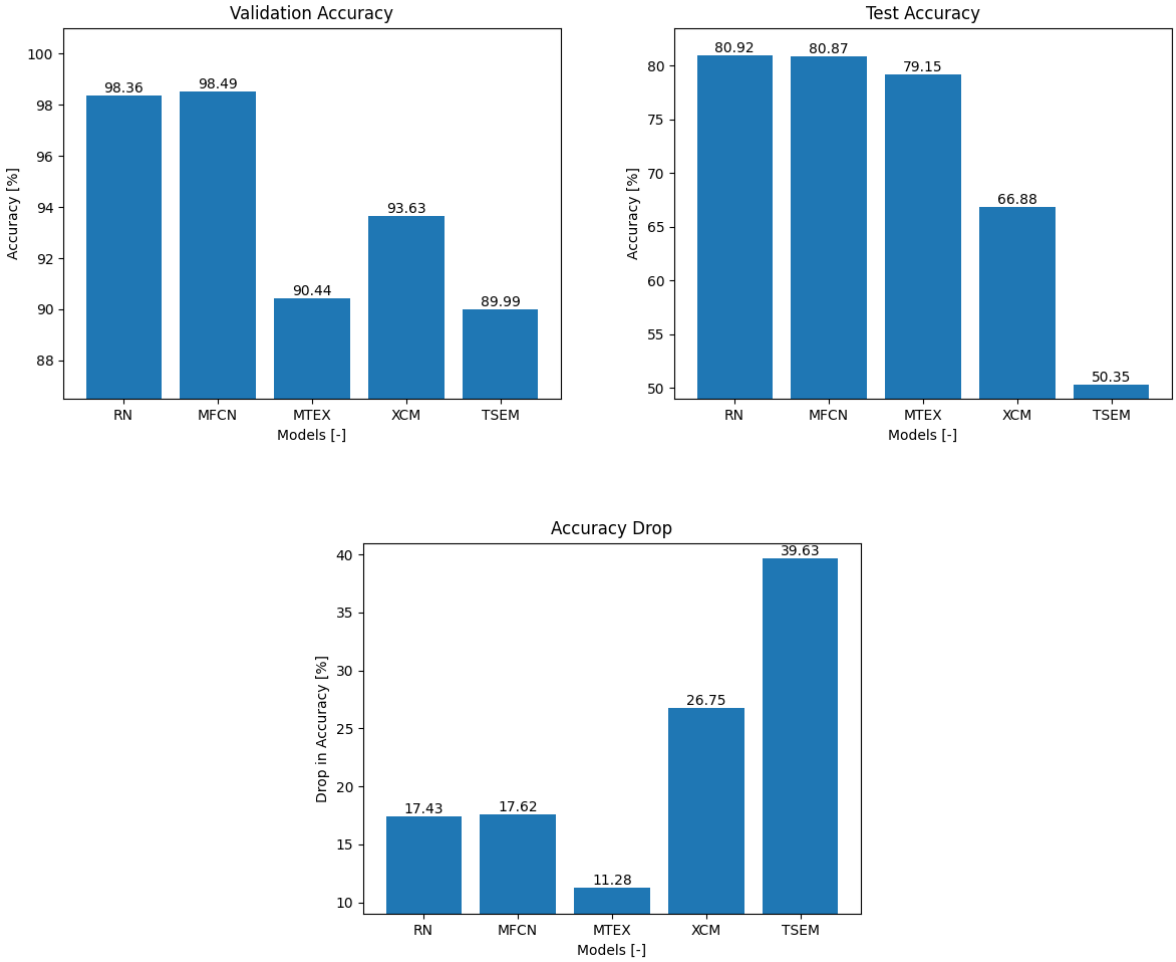

Figure 6.2: Results obtained by each model after 30 epochs. Abbreviations: *RN*—ResNet, *MLSTM*—MLSTM-FCN, *MTEX*—MTEX-CNN

It is also important to note that although all the models were trained for 30 epochs, due to their varying model structures, they all take different amounts of time to train. These training times are shown in Figure 6.3.

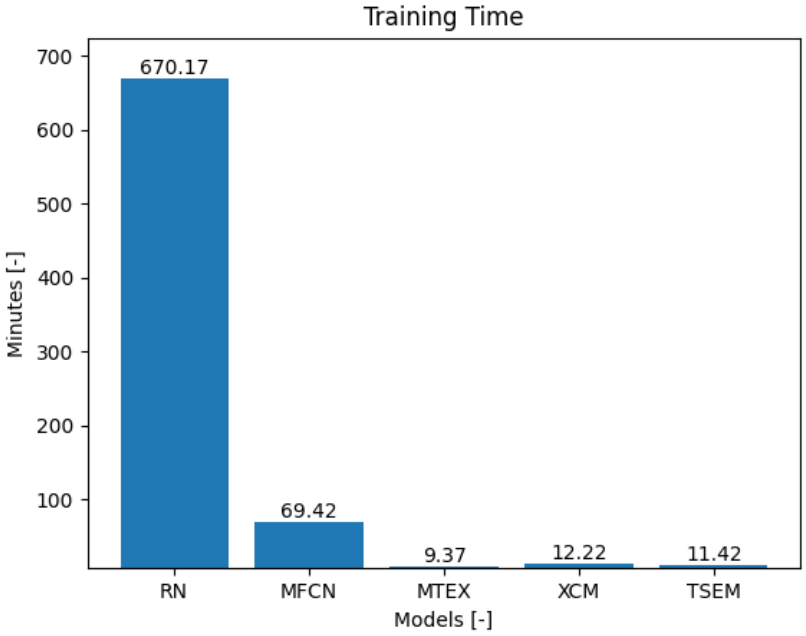

Figure 6.3: Training time for each model when trained for 30 epochs. Abbreviations: *RN*—ResNet, *MLSTM*—MLSTM-FCN, *MTEX*—MTEX-CNN

From this figure, it can be seen that there is a direct correlation between the time taken to train and the obtained validation accuracy. To determine whether or not the performance discrepancies found in the models were due to the imbalance in training time, these tests were performed again but instead of standardising the number of epochs the models will train with, the time taken to train will be standardised to 10 minutes. The results of the models with standardised training times are shown in the top left of Figure 6.4. Note that because a singular epoch for ResNet lasted 22.34 minutes, it is impossible to have the model run for 10 minutes. Therefore, the accuracies obtained from a singular performed epoch will be utilised.

As can be seen in the top left of Figure 6.4, with the lower training times, XCM now yields the highest validation accuracies at 93.22% with MLSTM-FCN, MTEX, and TSEM all having relatively similar accuracies at around 90%. The model which had the biggest difference in performance was ResNet which had a score of 81.85% which can be attributed to the fact that it ran for only one epoch. When obtaining their testing accuracies, which is shown in the top right of Figure 6.4, MTEX-CNN, XCM, and TSEM had the same relative trend as was obtained in top right of Figure 6.2 with MTEX obtaining the highest test accuracy, XCM obtaining a relatively lower test accuracy, and TSEM essentially labelling all time traces the same skill level, hence the 50% test accuracies. MLSTM-FCN and ResNet had significantly lower test accuracies compared to the ones obtained in top right Figure 6.2 with them now in between MTEX-CNN and XCM in terms of performance. However, when looking at the accuracy drops in the bottom of Figure 6.4, the same trend as shown in the bottom of Figure 6.2 appears. The only difference is regarding ResNet where it now obtains the lowest accuracy drop. However, this can be attributed to the low validation accuracy it obtained due to it running for one epoch.

Overall, by looking at all the results, the following models were dropped along with their reasoning:

- MLSTM-FCN: Between the two temporal analysing NN models, the MLSTM-FCN model was dropped. Although obtaining very similar validation and test accuracies to ResNet with significantly lower training time, MLSTM-FCN was dropped due to the fact that ResNet obtained the highest test accuracy when ran for 30 epochs out of all NN models.
- TSEM: Between the three variable and temporal analysing NN models, the TSEM model was dropped. Although producing reasonably high validation accuracies, the obtained test accuracies

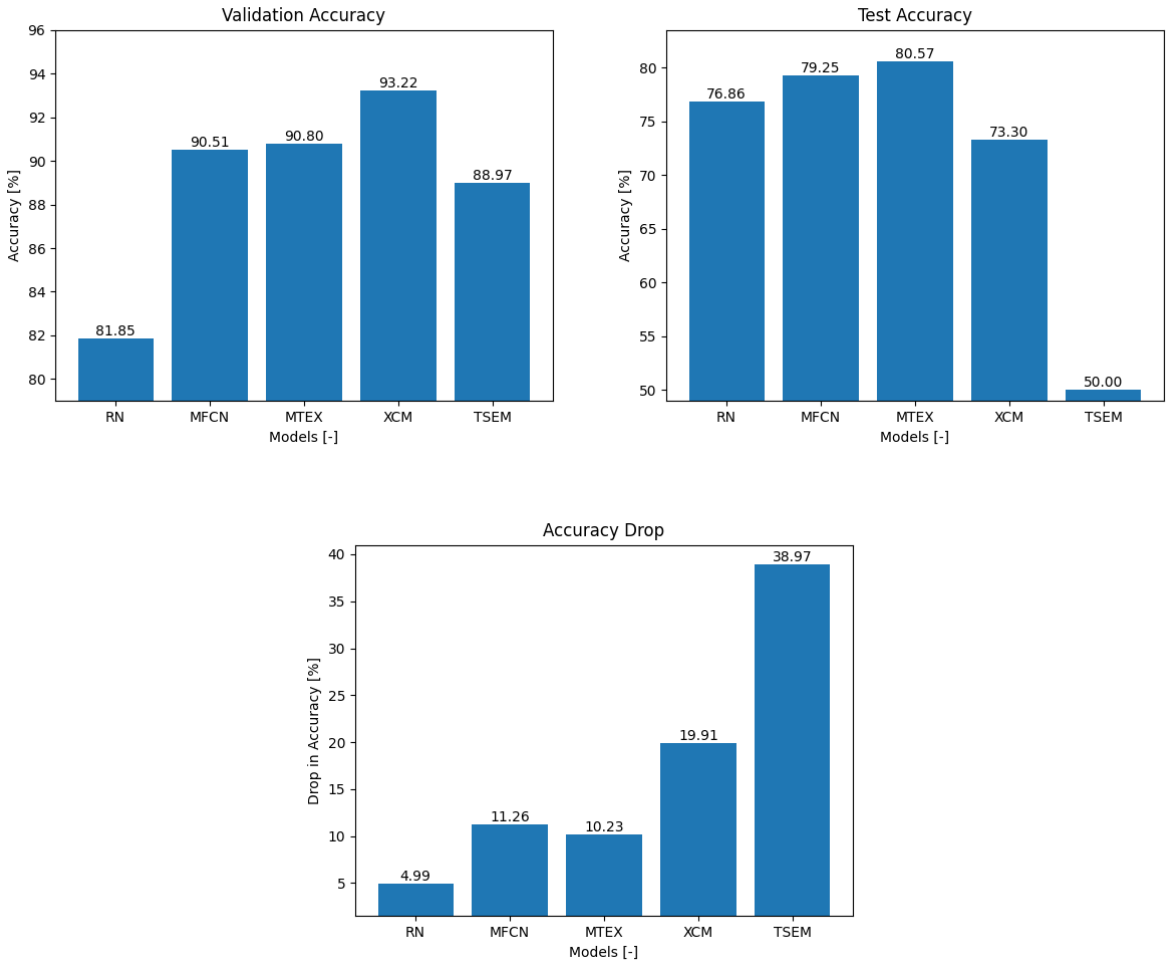

Figure 6.4: Results obtained by each model with 10 minutes of training time. Abbreviations: *RN*—ResNet, *MLSTM*—MLSTM-FCN, *MTEX*—MTEX-CNN

of 50% showed that the model was unable to pick up any underlying patterns from the presented tracking data and was labelling all the time traces as the same skill level.

Although XCM produced relatively low test accuracies, it was not dropped due to the fact that it has been shown to be able to produce more precise and accurate identification regarding which variables and time segments are most important in its classification [45]. Therefore, ResNet, MTEX-CNN, and XCM are the models that show the greatest promise in classifying the skill level of dual-axis tracking data whilst having the possibility of the addition of an explainability aspect. Overall, this implies that CNNs are most suitable for classifying tracking data with two different approaches being most successful, very deep CNNs such as ResNet and CNNs that run convolutional blocks extracting information from different dimensions such as MTEX-CNN and XCM.

6.4. XAI

With the most appropriate NNs identified, a preliminary implementation of XAI will be performed on the chosen models using the three Gradient-Based CAM methods that were discussed in Chapter 4, Grad-CAM, Grad-CAM++, and XGrad-CAM. To implement these Gradient-Based CAM methods, the Python package `TorchCAM` was used [92]. To showcase the utilisation of these CAM methods, a randomly chosen window of tracking data that has been correctly classified as 'unskilled' was used. The time traces of this window of tracking data are shown below in Figure 6.5.

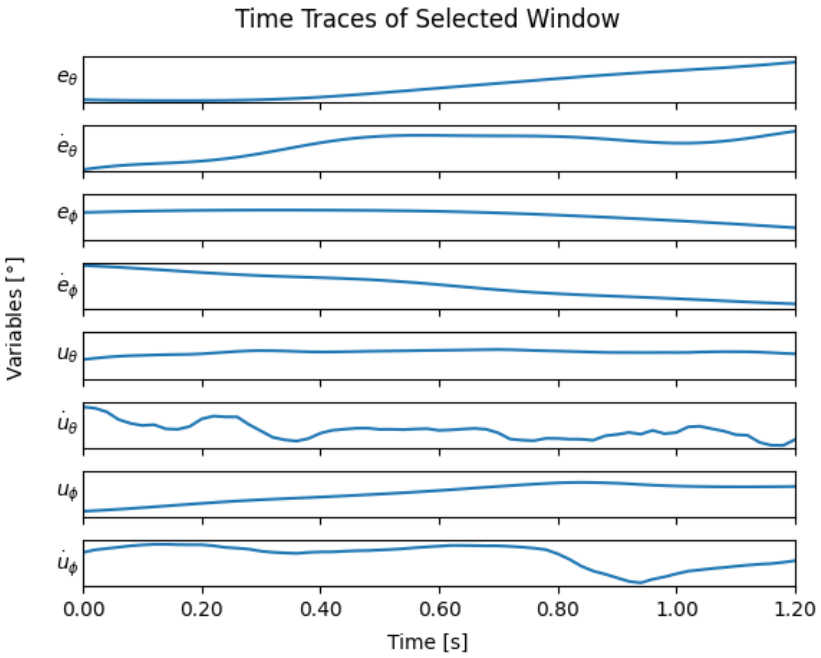

Figure 6.5: Time traces of randomly chosen window.

Note that although CAM produces class activation maps locally, that is per input, CAM can be used to produce the local class activation maps for all of the available tracking data in order to see a global trend regarding the most important details of the said data in order to be classified as either 'skilled' or 'unskilled', similar to what was done by de Jong when using SHAP [58, 66]. As this is currently out of the scope of the preliminary analysis presented in this report, this will be done in the main research.

6.4.1. ResNet

As was explained prior, ResNet simply analyses information relative to time in order to make its classification. Therefore, its class activation map is one-dimensional which highlights which time segments were most important in classifying the given input. Shown below are the class activation maps produced by each of the three aforementioned CAM methods for the randomly chosen window of tracking data.

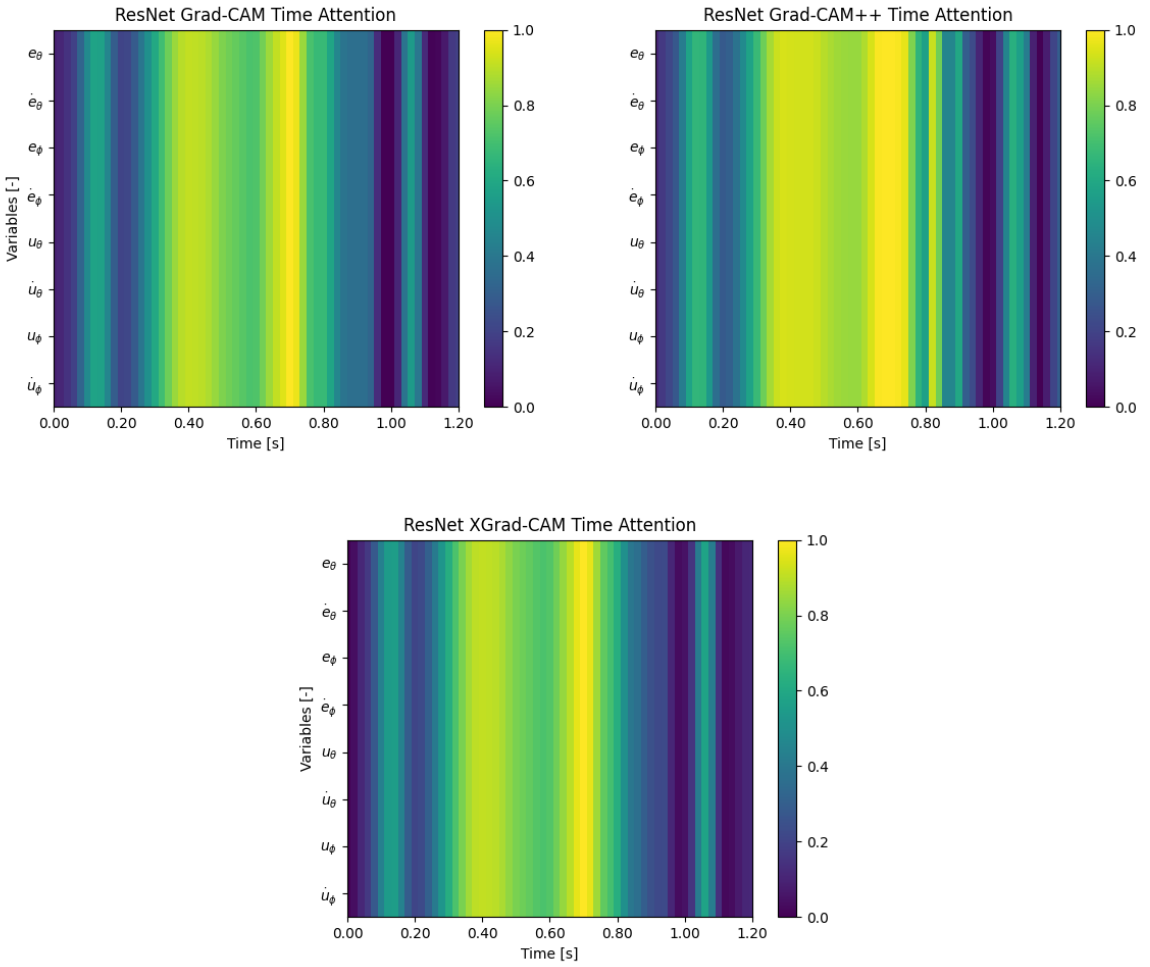

Figure 6.6: Class activation maps of ResNet after 30 epochs.

As can be seen, each of these CAM methods produced similar class activation maps regarding which time segment was most important in the NN model's classification choice. The class activation maps suggest that the time range between 0.30 [s] and 0.90 [s] was important in the model's classification with two time segments being most important: the time segment around 0.40 [s] and the time segment around 0.70 [s]. The time segment around 0.10 [s] and 1.10 [s] was also important in the classification made by the model, but not as much as those at 0.40 [s] and 0.70 [s]. However, due to ResNet's architecture, it is impossible to obtain a class activation map which represents which individual variables are most important in classifying the appropriate skill level, i.e. the lack of distinct variations along the y-axis. It can also be seen that the class activation maps produced by Grad-CAM and XGrad-CAM are extremely similar to one another compared to the one produced by Grad-CAM++. This can be attributed to the fact that, unlike Grad-CAm and XGrad-CAM, Grad-CAM++ utilises second-order gradients [82] whereas the others utilise first-order gradients [81, 83].

6.4.2. MTEX-CNN

Unlike ResNet, MTEX-CNN analyses information relative to both the variables of the input data and to time in order to make its classification. Therefore, for every input, two class activation maps can be obtained using CAM, one showing the most essential variables at certain times and one showing the most crucial time segments when taking into account all the variables. Similar to what was done with ResNet, shown in Figure 6.7 are the class activation maps produced by each of the three aforementioned CAM methods for the same randomly chosen window of tracking data.

As can be seen, when looking at the variable-wise class activation maps on the left side of Figure 6.7, no information was able to be extracted from the model regarding its classification. This inability of the CAM methods to extract the variable-wise class activation maps could be attributed to the fact that MTEX-CNN's network structure, as shown in Figure 4.20, has 2D and 1D convolution filters, which extract information relative to variables and time respectively, in sequence. As the 2D convolution filter responsible for extracting information relative to variables is the first layer out of four, the gradient at the first layer is essentially close to zero due to the vanishing gradient problem that affects deeper models [93]. Therefore, no information can be extracted from this layer. Note that although ResNet is a deeper NN, class activation maps were able to be extracted due to the fact that the layer from which the information was extracted was the very last layer of the model. In addition, ResNet, as was mentioned in Chapter 4, contains 'shortcuts' that circumvent the vanishing gradient problem by enabling gradient flow.

When looking at the temporal-wise class activation maps on the right side of Figure 6.7, information relative to the temporal dimension was able to be extracted. Similar to ResNet, this was possible due to the fact the 1D convolution filter responsible for extracting information relative to time is the very last layer of the model. It can be seen different time ranges were deemed important by the different CAM methods: time ranges 0.00 [s] to 0.60 [s] and 0.80 [s] to 1.20 [s] for Grad-CAM and time range 0.00 [s] to 1.00 [s] for XGrad-CAM. Additionally, it can also be seen that two different time segments were highlighted as most important by the different CAM methods, around 0.30 [s] for Grad-CAM and around 0.50 [s] for XGrad-CAM. This variability can be attributed to the fact that the convolutional filters in MTEX-CNN have no padding, which results in imprecise identification of time segments that are most important in classification as Grad-CAM is sensitive to upsampling processes caused by the lack of padding [45]. In contrast to Grad-CAM and XGrad-CAM, no information was extracted by Grad-CAM++. This can be attributed to Grad-CAM++'s reliance on second-order gradients [82] compared to Grad-CAM's and XGrad-CAM's reliance on first-order gradients [81, 83].

6.4.3. XCM

Similar to MTEX-CNN, XCM analyses information relative to both the variables of the input data and to time in order to make its classification. Therefore, shown in Figure 6.8 are the class activation maps produced by each of the three aforementioned CAM methods for the same randomly chosen window of tracking data.

When looking at the variable-wise class activation maps on the left side of Figure 6.8, it can be seen that although relatively dissimilar to one another, certain trends can be seen. The most apparent trend seen is the highlight of a singular variable at the very end of the window, u_ϕ for Grad-CAM and $\dot{e}_\phi$ for Grad-CAM++ and XGrad-CAM. Note that Grad-CAM does highlight $\dot{e}_\phi$ and both Grad-CAM++ and XGrad-CAM do highlight u_ϕ but not as brightly. The second trend seen is that the variables for the roll axis, ϕ, are

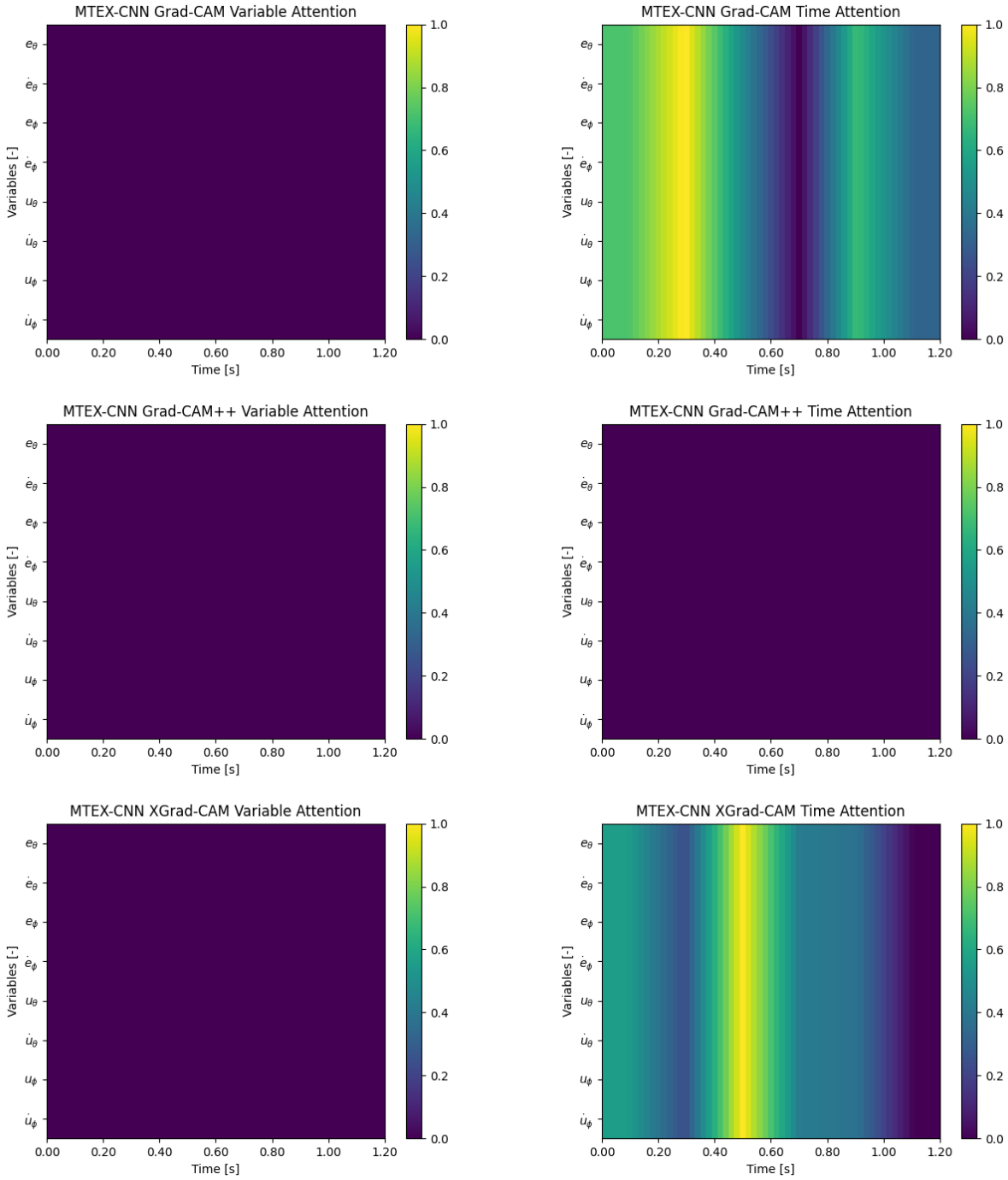

Figure 6.7: Class activation maps of MTEX-CNN after 30 epochs.

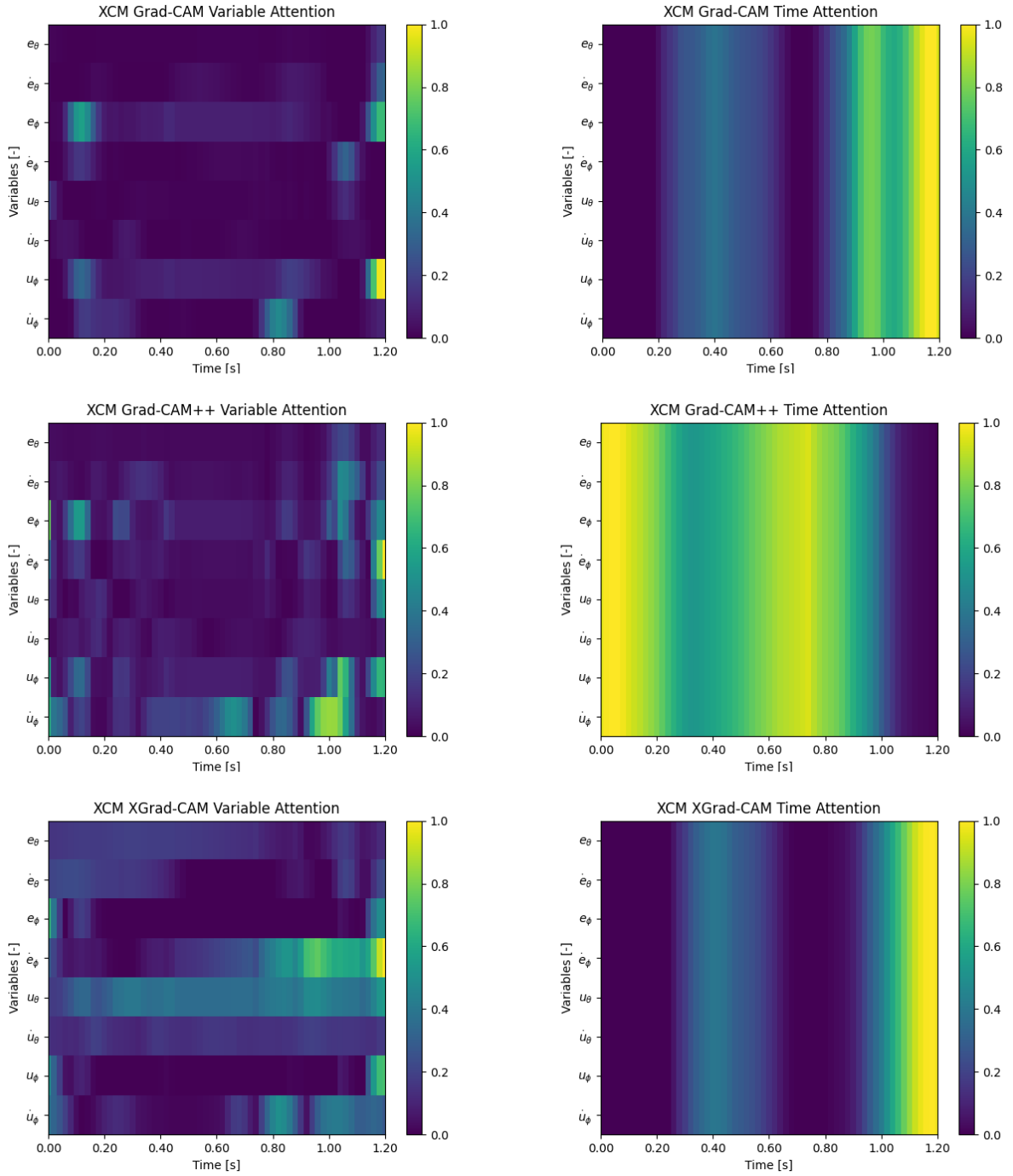

Figure 6.8: Class activation maps of XCM after 30 epochs.

highlighted relatively more by both Grad-CAM and Grad-CAM++. The third trend is the consistent dull highlighting of a few time segments of certain variables across the three CAM methods as shown in the highlight of e_ϕ and u_ϕ around 0.10 [s], $\dot{u}_\phi$ around 0.80 [s], $\dot{e}_\phi$ around 1.00 [s], and $\dot{e}_\theta$ around 1.20 [s].

When looking at the time-wise class activation maps on the right side of Figure 6.8, similar to what was seen with ResNet, the temporal class activation maps produced by Grad-CAM and XGrad-CAM are extremely similar to one another compared to the on produced by Grad-CAM++. The former highlights the time ranges 0.20 [s] to 0.60 [s] and 0.90 [s] to 1.20 [s] with the most important time segment around 1.20 [s]. In contrast, the latter appears to be the mirror image of the former with it highlighting the time range 0.00 [s] to 1.00 [s] with the most important time segments around 0.00 [s] and 0.70 [s]. This variability can again be attributed to Grad-CAM++'s reliance on second-order gradients [82] compared to Grad-CAM's and XGrad-CAM's reliance on first-order gradients [81, 83].

Overall, when looking at the obtained class activation maps from all the NN models, it can be seen that the class activation maps obtained from the three different Gradient-Based CAM methods are able to be used to verify these class activation maps against one another as they can share certain similarities. However, due to the possible variability that exists due to these different methods, another method of verification should be explored which would help further verify these class activation maps. One possible method would be the use of Activation Maximisation as used by de Jong [58]. As previously stated, Activation Maximisation [67] generates an input that maximises the neurons in the network for a certain class. Therefore, if this output resembles the global class activation map for the same class, it can provide evidence that the class activation maps are accurately highlighting the regions that the network is focusing on. Lastly, the class activation maps obtained from MTEX-CNN, or lack thereof, showcase how the network structure of a certain NN model has an impact regarding the possibility of adding an explainability aspect to said NN model.

6.5. RN-XCM

The chosen NN models, ResNet, MTEX-CNN, and XCM, all show great promise regarding classifying tracking data. However, each comes with its own major weakness that the other solves. ResNet displays extremely strong performance in the produced validation and testing accuracies. However, it does not allow for great interpretability as it cannot produce a variable-wise class activation map and takes a long time to train. MTEX-CNN display similarly high validation and testing accuracies to ResNet, though not as high, with significantly shorter training times. However, MTEX-CNN does not allow for great interpretability as its structure results in empty variable-wise class activation maps. XCM allows for great interpretability as it can produce both variable-wise and temporal-wise class activation maps while having relatively low training times. However, its obtained testing accuracy showcases the model's relatively poor performance in recognising the underlying patterns of the time traces. Therefore, in order to produce a model that shows exceptional performance whilst requiring low training time and allows for the production of both variable-wise and temporal-wise class activation maps, an additional test was performed on a new model, Residual Networks with Explainable Convolutional Neural Networks for Multivariate Time Series Classification (RN-XCM).

RN-XCM is a model that utilises XCM as its basis, but for the sections that produce a 1-dimensional feature map, a singular residual block from ResNet was utilised to create a deeper NN, though not as deep as ResNet. XCM was chosen as the basis of the model structure over MTEX-CNN due to the latter's inability to effectively produce variable-wise class activation maps due to its network structure. The network structure of RN-XCM is shown in Figure 6.9.

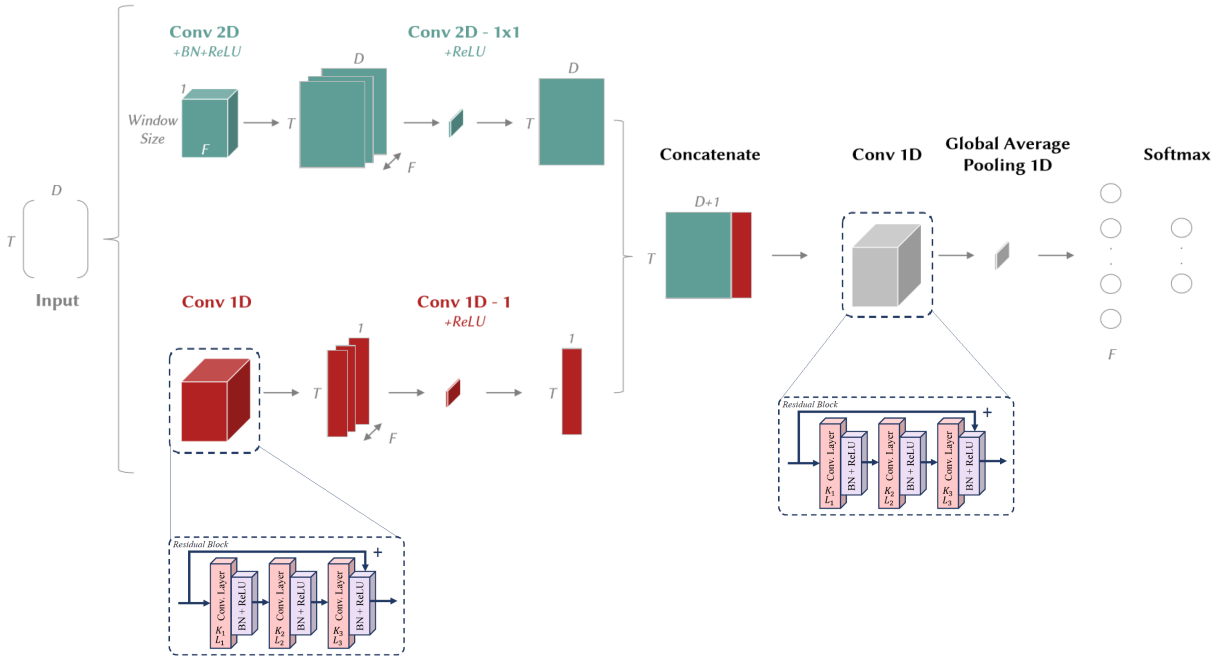

Figure 6.9: Network structure of RN-XCM, adapted from [45] and [58]. Abbreviations: *BN*—Batch Normalisation, *ReLU*—Rectified Linear Unit, *Window Size*—kernel size which corresponds to a percentage of *T*, *L*—kernel size which is predetermined as per ResNet, *D*—number of observed variables, *T*—time series length, *F*—number of feature maps, and *K*—number of feature maps within the Residual Blocks where the number of produced feature maps after them, *F*, equals K_3

6.5.1. Performance

In order to determine the viability of this model as a classifier of skill for human tracking data, the model underwent the same test the other models went through using the same hyperparameters shown in Table 6.2. The results can be seen in Figure 6.10 with RN-XCM shown in green. As can be seen, the inclusion of the deep residual block allowed for greater performance with RN-XCM clearly surpassing all the models in test accuracy and accuracy drop with its obtained validation accuracy coming second only to ResNet. In addition, due to the use of the convolutional blocks, it has a higher training time compared to MTEX-CNN and XCM though it is significantly faster than the training time required for ResNet.

Similar to what was done with the other NN models, the tests were reconducted with training time standardised to 10 minutes with the results shown in Figure 6.11. As can be seen, RN-XCM showcases high validation and test accuracies, with its test accuracy second only to MTEX-CNN.

Lastly, XAI was applied to the model using the same three Gradient-Based CAM methods with the class activation maps shown in Figure 6.12.

Starting with the variable attention class activation maps shown on the left side of Figure 6.12, it can be seen that unlike those obtained for XCM, obtained class activation maps are more consistent with one another. The most noticeable similarity between the three models is the fact the variables concerning the roll axis, ϕ, are highlighted much more than those concerning the pitch axis, θ. In addition, it can be seen that the variable attention class activation maps produced by Grad-CAM and XGrad-CAM are most similar to each other seen through the highlighting of variables e_ϕ and $\dot{u}_\phi$ at around 0.10 [s], variable u_ϕ at around 0.80 [s], and variables e_ϕ and u_ϕ at around 1.10 [s]. The variable attention class activation maps produced by Grad-CAM++ show some variation compared to the other methods as it highlights variable $\dot{u}_\phi$ at around 0.80 [s] instead of u_ϕ and variables e_ϕ and u_ϕ at around 1.20 [s] instead of at around 1.10 [s]. This variability can again be attributed to Grad-CAM's and XGrad-CAM's reliance on first-order gradients [81, 83] compared to Grad-CAM++'s reliance on second-order gradients [82].

When looking at the time attention class activation maps shown on the right side of Figure 6.12, it can be seen that they are far more consistent with one another, similar to those obtained by ResNet. When

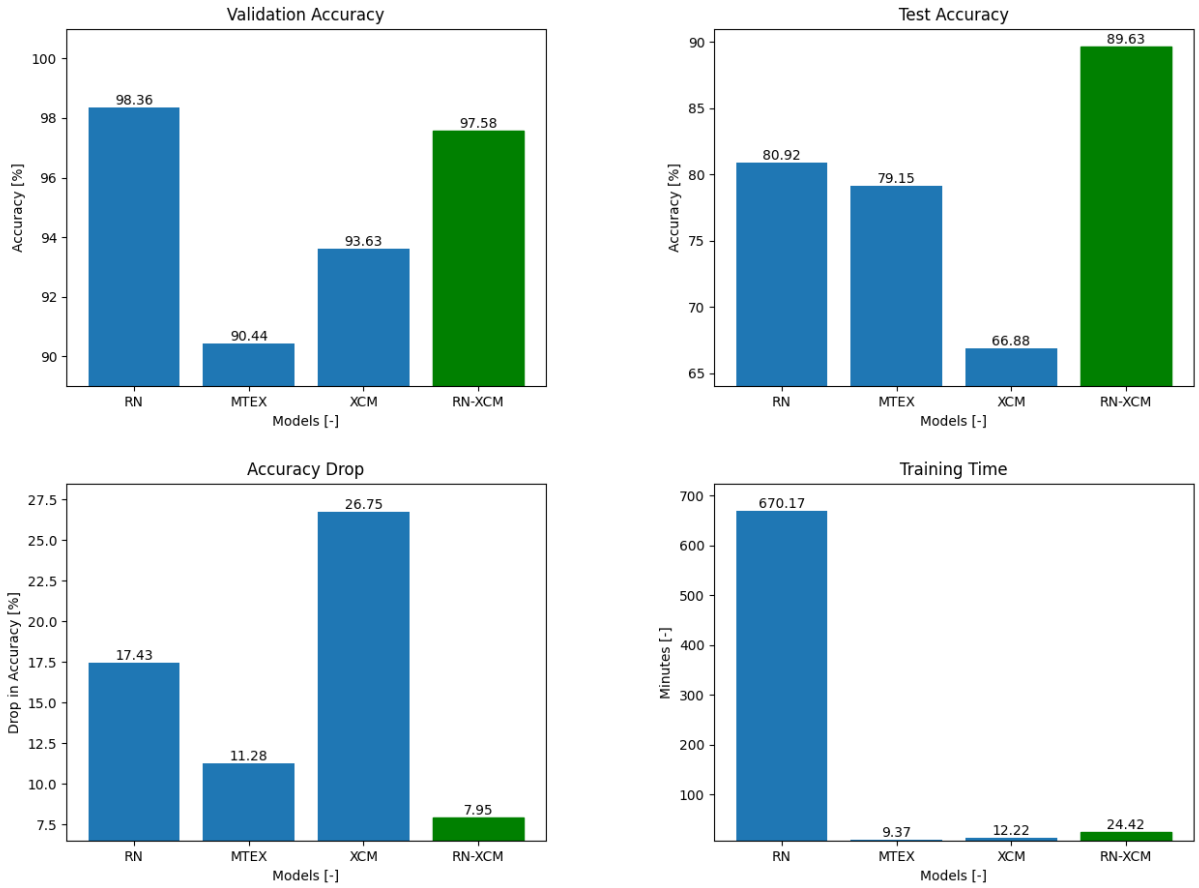

Figure 6.10: Results obtained by each model after 30 epochs. Abbreviations: Abbreviations: *RN*—ResNet, *MTEX*—MTEX-CNN

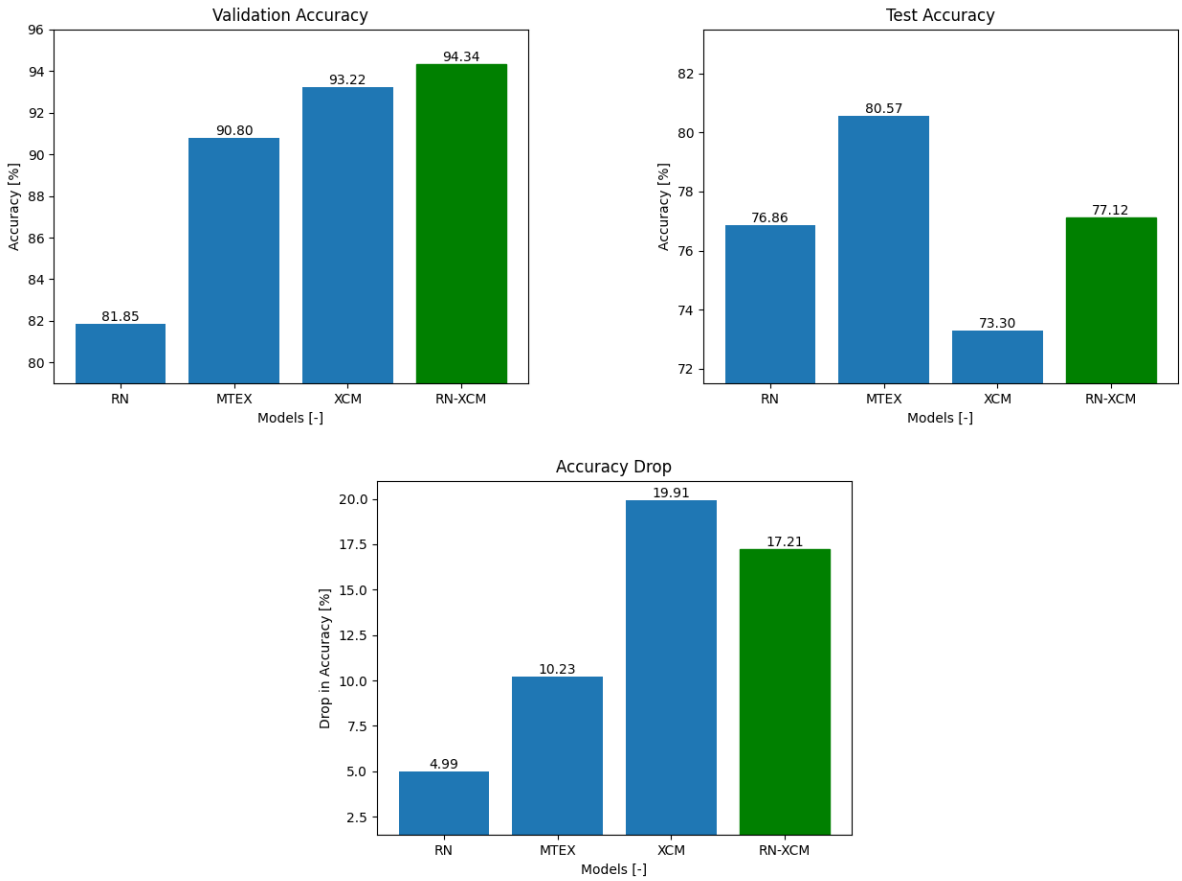

Figure 6.11: Results obtained by each model with 10 minutes of training time. Abbreviations: Abbreviations: *RN*—ResNet, *MTEX*—MTEX-CNN

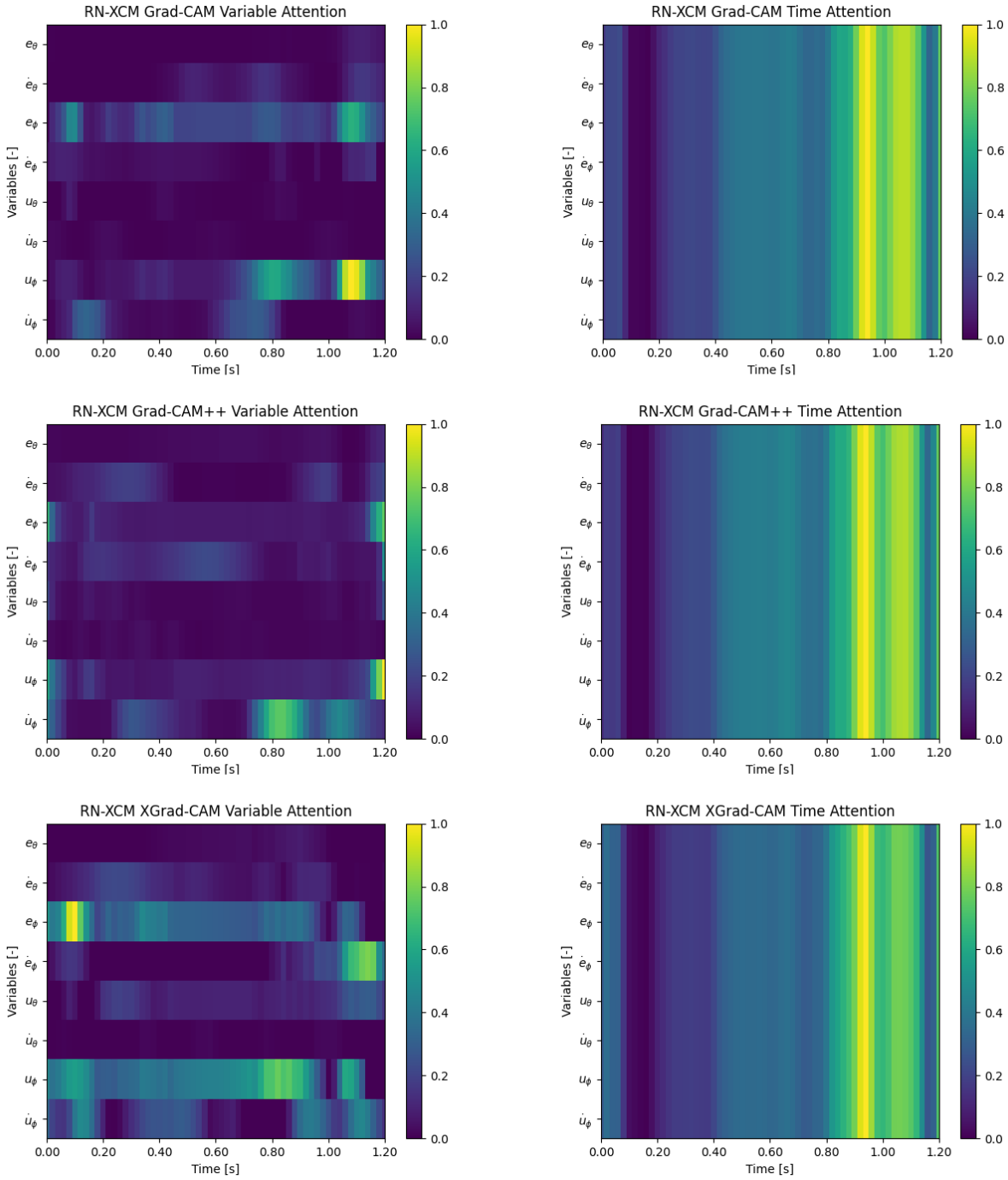

Figure 6.12: Class activation maps of RN-XCM after 30 epochs.

looking at these class activation maps, it can be seen that the entire time range was deemed important in classification with only the time segment around 0.10 [s] not important at all. In contrast, the two most important time segments are highlighted to be the time segment around 0.90 [s] and the time segment around 1.10 [s].

6.6. Key Takeaways

The chapter presented the rationale, methodology, and results of the preliminary test. The key takeaways of this chapter are as follows:

- When obtaining performance results for both a standardised amount of epochs and training time, ResNet, MTEX-CNN, and XCM yielded relatively the best overall performance. This suggests that two different model types yield the best performance for the current task, very deep networks and networks that analyse each dimension of the MTS data.
- Although these models show promising results, they contain significant drawbacks which the others excel. With this in mind, a new NN model was created, RN-XCM, which aims to outperform the three models in every aspect. When evaluating the performance of the newly synthesised model, the results showed that it indeed surpasses all the previous models in terms of obtained accuracies though with an increase in training time compared to MTEX-CNN and XCM, but still much faster than ResNet. The model's performance however makes up for this and will therefore be used for the main phase of the research.
- The use of CAM successfully allowed for insight regarding the logic and thought process behind the model classification of skill level with possible signs of understanding the cybernetic phenomena found in multi-axis tracking tasks. The use of three different Gradient-Based CAM methods allowed for proper verification of the obtained class activation maps as they showed variability with one another. However, a new XAI method should be looked into to allow for further verification. One possible candidate for this purpose is Activation Maximisation [67] which will be looked into in the main phase of the research.

7

Research Plan

In this chapter, the research plan will be established to ensure that all research sub-questions introduced in Chapter 5 are answered. To do so, the research will be split into three main parts.

7.1. Phase I: Optimise Neural Network Classifier

As was seen at the end of Chapter 6, RN-XCM showed great promise regarding properly classifying tracking runs it has both seen and never seen whilst also providing great insight into its logic. This has successfully answered **Sub-question 1(a)**.

Similar to XCM, RN-XCM contains a model-specific hyperparameter, the window size hyperparameter, which needs to be optimised. Therefore, this hyperparameter shall be optimised heuristically to answer **Sub-question 1(b)**. The heuristic optimisation of input signals will also be done to answer **Sub-question 1(c)**.

Due to the number of values to test for the model-specific hyperparameter alongside the determination of the ideal combination of input signals, this phase will take up an estimated amount of 2 weeks.

7.2. Phase II: Assess Model Robustness

In this phase, the performance of the classifiers will be assessed fully by using the entirety of tracking runs for both training and validation. This will answer **Sub-question 2(a)**. Afterwards, the performance of the classifiers with never before seen data will be assessed by removing one subject at a time during the training phase. The average performance after removing each participant will be used to assess the classifiers and will answer **Sub-question 2(b)**. In addition, the performance of the model on single-axis tracking data in order to answer **Sub-question 2(c)**. The single-axis tracking data will come from the tracking data of the dual-axis tracking data used for this research [30] and isolate it to either only the pitch or the roll axis. This is done to determine if the model performs better when trained and evaluated with multi-axis data compared to either pitch-only single-axis or roll-only single-axis data. If so, this would infer that the model understands the phenomena that occur due to multi-axis tracking tasks as shown in [23].

Due to the number of subjects available in determining the performance of the model, it will extend the time required to complete this phase. Therefore, this phase is expected to take 3 weeks.

7.3. Phase III: Determine Model Comprehensiveness

In this phase, Grad-CAM, Grad-CAM++, and XGrad-CAM will be used to interpret the results produced by the classifiers in Phase II. It will also be used to determine whether both classifiers interpret the model in the same manner to classify tracking runs. This will answer **Sub-questions 3(a)** and **3(b)**.

Due to the fact that the XAI methods were already implemented for the preliminary analysis in this report, the phase is expected to take no longer than 1 week.

8

Conclusion

As stated in the introduction, the objective of this research is to produce a Deep Learning model which would be able to classify a HO's skill level based on their control strategy of a two-axis tracking task in real-time. This preliminary report was the first step in doing so as it explored the topics relevant to the goal of this research such that these topics could be used together effectively within the main phase of the research. As this preliminary report consisted of multiple phases, the following paragraphs will conclude how each of them aided in fulfilling the research objective.

Literature Study: The literature study phase was conducted to understand the most state-of-the-art knowledge regarding the classification of skill level for human behaviour in dual axis tracking tasks within two fields, cybernetics and deep learning.

Regarding the literature study for cybernetics, it was found that although it was possible to assess human skill levels through the use of the Crossover theory, it produces results that do fully capture the non-linear and time-invariant nature of human control strategies, but is fundamentally unfit to provide any real-time feedback. It was also found that although there are methods that allow for the identification of time-varying control behaviour, most do not allow for online usage. In addition, models that do allow for online usage simply are not accurate enough for reliable usage. Within the realm of cybernetics, it was discovered that the introduction of more than a single axis in a tracking introduces four distinct phenomena, performance degradation, axis asymmetry, crossfeed, and intermittency. It is of interest to determine whether or not the proposed NN model can recognise these phenomena. Lastly, it was found that the remnant signal had a significant contribution in skill-acquisition, specifically higher contribution in earlier runs, runs with lower skill levels, and lower contribution in later runs, runs with higher skill levels. The proposed NN model would therefore need to be able to identify the non-linear and time-varying remnant signal in order to have strong classification performance.

Regarding the literature study of deep learning, it was found that two distinct NN architectures are predominantly used in MTSC, CNNs and RNNs. In addition, it was found that a similar study that used a deep NN model that heavily relied on CNNs, ResNet, to classify the skill level of human control strategies on single-axis tracking found success as the model obtained extremely high validation accuracies. However, there was a performance drop when obtaining the test accuracies, which mimicked its possible real-world usage, which was deduced to be caused by the complexity that came with classifying skill level. This study also explored the possibility of adding an explainability aspect to allow for greater insight into what aspects of the tracking data is most responsible for classifying skill level. Although the use of SHAP and Activation Maximisation within the study yielded promising results, the research discussed its limitations due to the fact that it was unable to extract the logic within the classifier when making its classification. When selecting models for MTSC, three selection criteria were established: performance, ease of implementation, and complexity. The use of these criteria led to five models being used for the preliminary testing phase of the research: ResNet, MSLTM-FCN, MTEX-CNN, XCM, and TSEM. Lastly, when determining the best method to allow for an explainability aspect for the developing model, it was found CAM was ideal due to its applicability and ability to extract the NN model's thought process in making its classifications.

Preliminary Analysis: During the preliminary analysis phase of the literature study, the aforementioned five NN models were tested with both standardised epochs and training times in obtaining both validation accuracies and testing accuracies. When evaluating the results, two models showed the greatest promise:

ResNet and XCM. CAM was then used to understand the logic behind the two models. Overall, it was observed that both models contained weaknesses at which the other excels. Therefore, both models were combined to obtain a fully new model, RN-XCM. This new model displayed exceptional performance in validation and testing accuracies whilst maintaining relatively short training times and allowing for exceptional explainability when CAM was applied. Therefore, in the main phase of the research, only RN-XCM will be utilised.

Main Phase Planning: During the main phase of the research, research questions stemming from the literature study will be answered. To allow for structure execution, the main phase will be split into three main phases. The first phase aims to build the most optimal NN classifier by optimising the NN hyperparameters and data pre-processing steps. The second phase aims to determine the model's overall robustness in ideal and real-world scenarios. Lastly, the third phase aims to determine the feasibility of CAM as an XAI method and determine what characteristics of human control strategy of dual-axis tracking tasks result in an 'unskilled' or 'skilled' classification.

References

[1] D. M. Pool et al. “A Cybernetic Approach to Assess the Training of Manual Control Skills”. In: *IFAC-PapersOnLine* 49.19 (2016). 13th IFAC Symposium on Analysis, Design, and Evaluation ofHuman-Machine Systems HMS 2016, pp. 343–348. DOI: https://doi.org/10.1016/j.ifacol.2016.10.588. URL: https://www.sciencedirect.com/science/article/pii/S2405896316321796.

[2] M. Mulder et al. “Manual Control Cybernetics: State-of-the-Art and Current Trends”. In: *IEEE Transactions on Human-Machine Systems* 48.5 (2018), pp. 468–485. DOI: 10.1109/THMS.2017.2761342.

[3] P. P. Shinde et al. “A Review of Machine Learning and Deep Learning Applications”. In: *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*. 2018, pp. 1–6. DOI: 10.1109/ICCUBEA.2018.8697857.

[4] J. Rasmussen. “Skills, rules, and knowledge; signals, signs, and symbols, and other distinctions in human performance models”. In: *IEEE Transactions on Systems, Man, and Cybernetics* SMC-13.3 (1983), pp. 257–266. DOI: 10.1109/TSMC.1983.6313160.

[5] D. T. McRuer. “HUMAN PILOT DYNAMICS IN COMPENSATORY SYSTEMS”. In: 1965.

[6] M. Mulder et al. “AE4316 Aerospace Human-Machine Systems”. 2022.

[7] D.T. McRuer et al. “A Review of Quasi-Linear Pilot Models”. In: *IEEE Transactions on Human Factors in Electronics* HFE-8.3 (1967), pp. 231–249. DOI: 10.1109/THFE.1967.234304.

[8] M. Mendes et al. “Effects of Peripheral Visual Cues in Simulator-Based Training of Multimodal Control Skills”. In: June 2017. DOI: 10.2514/6.2017-3671.

[9] L. R. Young. “On Adaptive Manual Control”. In: *IEEE Transactions on Man-Machine Systems* 10.4 (1969), pp. 292–331. DOI: 10.1109/TMMS.1969.299931.

[10] A. Karniel et al. “Human motor control: learning to control a time-varying, nonlinear, many-to-one system”. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 30.1 (2000), pp. 1–11. DOI: 10.1109/5326.827449.

[11] P. M. T. Zaal et al. “Modeling Human Multimodal Perception and Control Using Genetic Maximum Likelihood Estimation”. In: *Journal of Guidance, Control, and Dynamics* 32.4 (2009), pp. 1089–1099. DOI: 10.2514/1.42843. eprint: https://doi.org/10.2514/1.42843. URL: https://doi.org/10.2514/1.42843.

[12] Peter Zaal et al. “Estimation of Time-Varying Pilot Model Parameters”. In: *AIAA Modeling and Simulation Technologies Conference*. DOI: 10.2514/6.2011-6474. eprint: https://arc.aiaa.org/doi/pdf/10.2514/6.2011-6474. URL: https://arc.aiaa.org/doi/abs/10.2514/6.2011-6474.

[13] David Klyde et al. “Wavelet-based time-varying human operator models”. In: *AIAA Atmospheric Flight Mechanics Conference and Exhibit*. DOI: 10.2514/6.2001-4009. eprint: https://arc.aiaa.org/doi/pdf/10.2514/6.2001-4009. URL: https://arc.aiaa.org/doi/abs/10.2514/6.2001-4009.

[14] Mario Olivari et al. “Identifying Time-Varying Pilot Responses: a Regularized Recursive Least-Squares Algorithm”. In: Jan. 2016. DOI: 10.2514/6.2016-1182.

[15] Mario Olivari et al. “Identifying time-varying neuromuscular system with a recursive least-squares algorithm: a Monte-Carlo simulation study”. In: *2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. 2014, pp. 3573–3578. DOI: 10.1109/SMC.2014.6974484.

[16] James R. Schiess et al. “Kalman filter estimation of human pilot-model parameters”. In: 1975.

[17] Alexandru Popovici et al. “Dual Extended Kalman Filter for the Identification of Time-Varying Human Manual Control Behavior”. In: *AIAA Modeling and Simulation Technologies Conference*. DOI: 10.

2514/6.2017-3666. eprint: https://arc.aiaa.org/doi/pdf/10.2514/6.2017-3666. URL: https://arc.aiaa.org/doi/abs/10.2514/6.2017-3666.

[18] Jim Rojer et al. “UKF-based Identification of Time-Varying Manual Control Behaviour”. In: *IFAC-PapersOnLine* 52.19 (2019). 14th IFAC Symposium on Analysis, Design, and Evaluation of Human Machine Systems HMS 2019, pp. 109–114. DOI: https://doi.org/10.1016/j.ifacol.2019.12.120. URL: https://www.sciencedirect.com/science/article/pii/S240589631931955X.

[19] Alexandru Popovici et al. “Time-Varying Manual Control Identification in a Stall Recovery Task under Different Simulator Motion Conditions”. In: June 2018. DOI: 10.2514/6.2018-2936.

[20] E.R. Boer et al. “Estimation of time-varying delay time in nonstationary linear systems: an approach to monitor human operator adaptation in manual tracking tasks”. In: *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans* 28.1 (1998), pp. 89–99. DOI: 10.1109/3468.650325.

[21] W. Plaetinck et al. “Online Identification of Pilot Adaptation to Sudden Degradations in Vehicle Stability”. In: vol. 51. Dec. 2018, pp. 347–352. DOI: 10.1016/j.ifacol.2019.01.020.

[22] Andries van Grootheest et al. “Identification of Time-Varying Manual Control Adaptations with Recursive ARX Models”. In: *2018 AIAA Modeling and Simulation Technologies Conference*. DOI: 10.2514/6.2018-0118. eprint: https://arc.aiaa.org/doi/pdf/10.2514/6.2018-0118. URL: https://arc.aiaa.org/doi/abs/10.2514/6.2018-0118.

[23] S. Barendswaard et al. “Dual-Axis Manual Control: Performance Degradation, Axis Asymmetry, Crossfeed, and Intermittency”. In: *IEEE Transactions on Human-Machine Systems* 49.2 (2019), pp. 113–125. DOI: 10.1109/THMS.2019.2890856.

[24] H. P. Berger et al. *The Effects of Motion Cues and Motion Scaling on One- and Two-axis Compensatory Control Tasks*. National Aeronautics and Space Administration, 1971.

[25] J. Adams. “Human Transfer Functions for Multi-Axis and Multi-Loop Problems”. In: *4th Manned Space Flight Meeting*. DOI: 10.2514/6.1965-1521. eprint: https://arc.aiaa.org/doi/pdf/10.2514/6.1965-1521. URL: https://arc.aiaa.org/doi/abs/10.2514/6.1965-1521.

[26] R. A. Hess. “Modeling Human Pilot Adaptation to Flight Control Anomalies and Changing Task Demands”. In: *Journal of Guidance, Control, and Dynamics* 39.3 (2016), pp. 655–666. DOI: 10.2514/1.G001303. eprint: https://doi.org/10.2514/1.G001303. URL: https://doi.org/10.2514/1.G001303.

[27] G. A. Bekey et al. “Mathematical models of human operators in simple two-axis manual control systems”. In: *IEEE Transactions on Human Factors in Electronics* HFE-6.1 (1965), pp. 42–52. DOI: 10.1109/THFE.1965.6591255.

[28] David Mitchell et al. “Minimum Flying Qualities. Volume 1. Piloted Simulation Evaluation of Multiple Axis Flying Qualities”. In: (Jan. 1990), p. 447.

[29] D. M. Pool et al. “A Cybernetic Approach to Assess Flight Simulator Motion Fidelity”. In: *IFAC Proceedings Volumes* 43.13 (2010). 11th IFAC/IFIP/IFORS/IEA Symposium on Analysis, Design, and Evaluation of Human-Machine Systems, pp. 380–385. DOI: https://doi.org/10.3182/20100831-4-FR-2021.00067. URL: https://www.sciencedirect.com/science/article/pii/S1474667015325623.

[30] R. Wijlens et al. “Retention of Manual Control Skills in Multi-Axis Tracking Tasks”. In: Jan. 2020. DOI: 10.2514/6.2020-2264.

[31] B. A. van Leeuwen et al. “Prediction Models for Individuals’ Control Skill Development and Retention using XGBoost and SHAP”. In: *AIAA SCITECH 2023 Forum*. DOI: 10.2514/6.2023-0542. eprint: https://arc.aiaa.org/doi/pdf/10.2514/6.2023-0542. URL: https://arc.aiaa.org/doi/abs/10.2514/6.2023-0542.

[32] Molly Ruby. *How ChatGPT Works: The Model Behind The Bot*. Accessed on March, 2023. 2023. URL: https://towardsdatascience.com/how-chatgpt-works-the-models-behind-the-bot-1ce5fca96286.

[33] Josh Patterson et al. *Deep learning: A practitioner□s approach*. O□Reilly, 2017.

[34] F. Bre et al. “Prediction of wind pressure coefficients on building surfaces using artificial neural networks”. In: *Energy and Buildings* 158 (2018), pp. 1429–1441. DOI: https://doi.org/10.1016/j.enbuild.2017.11.045. URL: https://www.sciencedirect.com/science/article/pii/S0378778817325501.

[35] Ian J. Goodfellow et al. *Deep Learning*. http://www.deeplearningbook.org. Cambridge, MA, USA: MIT Press, 2016.

[36] Jeremy Jordan. *Setting the learning rate of your neural network.* Accessed on March, 2023. 2018. URL: https://www.jeremyjordan.me/nn-learning-rate/.

[37] Raoof Naushad et al. “Deep Transfer Learning for Land Use and Land Cover Classification: A Comparative Study”. In: *Sensors* 21 (Dec. 2021), p. 8083. DOI: 10.3390/s21238083.

[38] Nitish Srivastava et al. “Dropout: A Simple Way to Prevent Neural Networks from Overfitting”. In: *Journal of Machine Learning Research* 15.56 (2014), pp. 1929–1958. URL: http://jmlr.org/papers/v15/srivastava14a.html.

[39] Brian Kenji Iwana et al. “An empirical survey of data augmentation for time series classification with neural networks”. In: *PLOS ONE* 16.7 (July 2021). Ed. by Friedhelm Schwenker, e0254841. DOI: 10.1371/journal.pone.0254841. URL: https://doi.org/10.1371%5C%2Fjournal.pone.0254841.

[40] Alejandro Pasos Ruiz et al. “The Great Multivariate Time Series Classification Bake off: A Review and Experimental Evaluation of Recent Algorithmic Advances”. In: *Data Min. Knowl. Discov.* 35.2 (Mar. 2021), pp. 401–449. DOI: 10.1007/s10618-020-00727-3. URL: https://doi.org/10.1007/s10618-020-00727-3.

[41] Zhiguang Wang et al. *Time Series Classification from Scratch with Deep Neural Networks: A Strong Baseline*. 2016. DOI: 10.48550/ARXIV.1611.06455. URL: https://arxiv.org/abs/1611.06455.

[42] Hassan Ismail Fawaz et al. “InceptionTime: Finding AlexNet for time series classification”. In: *Data Mining and Knowledge Discovery* 34.6 (Sept. 2020), pp. 1936–1962. DOI: 10.1007/s10618-020-00710-y. URL: https://doi.org/10.1007%5C%2Fs10618-020-00710-y.

[43] Fazle Karim et al. “Multivariate LSTM-FCNs for time series classification”. In: *Neural Networks* 116 (Aug. 2019), pp. 237–245. DOI: 10.1016/j.neunet.2019.04.014. URL: https://doi.org/10.1016%5C%2Fj.neunet.2019.04.014.

[44] Roy Assaf et al. “MTEX-CNN: Multivariate Time Series EXplanations for Predictions with Convolutional Neural Networks”. In: *2019 IEEE International Conference on Data Mining (ICDM)*. 2019, pp. 952–957. DOI: 10.1109/ICDM.2019.00106.

[45] Kevin Fauvel et al. “XCM: An Explainable Convolutional Neural Network for Multivariate Time Series Classification”. In: *Mathematics* 9.23 (Dec. 2021), p. 3137. DOI: 10.3390/math9233137. URL: https://doi.org/10.3390%5C%2Fmath9233137.

[46] Anh-Duy Pham et al. *TSEM: Temporally Weighted Spatiotemporal Explainable Neural Network for Multivariate Time Series*. 2022. DOI: 10.48550/ARXIV.2205.13012. URL: https://arxiv.org/abs/2205.13012.

[47] Samira Pouyanfar et al. “A Survey on Deep Learning: Algorithms, Techniques, and Applications”. In: *ACM Comput. Surv.* 51.5 (Sept. 2018). DOI: 10.1145/3234150. URL: https://doi.org/10.1145/3234150.

[48] Keiron O’Shea et al. *An Introduction to Convolutional Neural Networks*. 2015. DOI: 10.48550/ARXIV.1511.08458. URL: https://arxiv.org/abs/1511.08458.

[49] Kenneth Lee. *A Theory Explains Deep Learning*. Accessed on March, 2023. 2017. URL: https://medium.com/@kennethlee_98497/a-theory-explains-deep-learning-79b02598a2d5.

[50] Aurélien Géron. *Hands-on machine learning with scikit-learn, Keras, and tensorflow: Concepts, tools, and techniques to build Intelligent Systems*. O’Reilly, 2020.

[51] NVIDIA. *Convolutional Neural Network (CNN)*. Accessed on March, 2023. URL: https://developer.nvidia.com/discover/convolutional-neural-network.

[52] Derrick Mwiti. *Getting Started with Recurrent Neural Network (RNNs)*. Accessed on March, 2023. 2022. URL: https://towardsdatascience.com/getting-started-with-recurrent-neural-network-rnns-ad1791206412.

[53] Fatemeh Madaeni et al. "Convolutional Neural Network and Long Short-Term Memory Models for Ice-Jam Prediction". In: (July 2021). DOI: 10.5194/tc-2021-194.

[54] Sepp Hochreiter et al. "Long Short-term Memory". In: *Neural computation* 9 (Dec. 1997), pp. 1735–80. DOI: 10.1162/neco.1997.9.8.1735.

[55] Wenjie Lu et al. "A CNN-LSTM-based model to forecast stock prices". In: *Complexity* 2020 (Nov. 2020), pp. 1–10. DOI: 10.1155/2020/6622927.

[56] R. Versteeg. "Classifying Human Control Behaviour by Artificial Intelligence". MA thesis. 2019. URL: http://resolver.tudelft.nl/uuid:9adc6dfe-5f34-4936-a4d8-c0d35df7ee37.

[57] G.J.H.A. Verkerk. "Classifying Human Manual Control Behavior in Tracking Tasks with Various Display types Using the InceptionTime CNN". MA thesis. 2021. URL: http://resolver.tudelft.nl/uuid:74e19a71-01b9-4ce5-b684-15709771f1f7.

[58] M.J.L. de Jong. "Classifying Human Pilot Skill Level Using Deep Artificial Neural Networks". MA thesis. 2021. URL: http://resolver.tudelft.nl/uuid:f09da3e9-3220-46c4-9a3e-e066c9fb3ea0.

[59] D. M. Pool et al. "Effects of Simulator Motion Feedback on Training of Skill-Based Control Behavior". In: *Journal of Guidance, Control, and Dynamics* 39.4 (2016), pp. 889–902. DOI: 10.2514/1.G001603. eprint: https://doi.org/10.2514/1.G001603. URL: https://doi.org/10.2514/1.G001603.

[60] Mateusz Buda et al. "A systematic study of the class imbalance problem in convolutional neural networks". In: *Neural Networks* 106 (Oct. 2018), pp. 249–259. DOI: 10.1016/j.neunet.2018.07.011. URL: https://doi.org/10.1016%5C%2Fj.neunet.2018.07.011.

[61] Justin Johnson et al. "Survey on deep learning with class imbalance". In: *Journal of Big Data* 6 (Mar. 2019), p. 27. DOI: 10.1186/s40537-019-0192-5.

[62] Charles Stark Draper et al. "The roles of men and instruments in control and guidance systems for spacecraft". In: *15th International Astronautical Congress, Poland*. 1964.

[63] Albert E. Preyss et al. "Stochastic Modeling of Human Learning Behavior". In: *IEEE Transactions on Man-Machine Systems* 9.2 (1968), pp. 36–46. DOI: 10.1109/TMMS.1968.300006.

[64] Yann A. LeCun et al. "Efficient BackProp". In: *Neural Networks: Tricks of the Trade: Second Edition*. Ed. by Grégoire Montavon et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 9–48. DOI: 10.1007/978-3-642-35289-8_3. URL: https://doi.org/10.1007/978-3-642-35289-8_3.

[65] Martijn J.L. de Jong et al. "Cybernetic Data Augmentation for Neural Network Classification of Control Skills". In: *IFAC-PapersOnLine* 55.29 (2022). 15th IFAC Symposium on Analysis, Design and Evaluation of Human Machine Systems HMS 2022, pp. 178–183. DOI: https://doi.org/10.1016/j.ifacol.2022.10.252. URL: https://www.sciencedirect.com/science/article/pii/S2405896322022790.

[66] Scott M Lundberg et al. "A Unified Approach to Interpreting Model Predictions". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper/2017/file/8a20a8621978632d76c43dfd28b67767-Paper.pdf.

[67] D. Erhan et al. "Visualizing Higher-Layer Features of a Deep Network". In: 2009.

[68] Karen Simonyan et al. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2014. DOI: 10.48550/ARXIV.1409.1556. URL: https://arxiv.org/abs/1409.1556.

[69] Xuchao Zhang et al. "TapNet: Multivariate Time Series Classification with Attentional Prototypical Network". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 34.04 (Apr. 2020),

pp. 6845–6852. DOI: 10.1609/aaai.v34i04.6165. URL: https://ojs.aaai.org/index.php/AAAI/article/view/6165.

[70] Seyed Navid Mohammadi Foumani et al. “Disjoint-CNN for Multivariate Time Series Classification”. In: *2021 International Conference on Data Mining Workshops (ICDMW)*. 2021, pp. 760–769. DOI: 10.1109/ICDMW53433.2021.00099.

[71] Guozhong Li et al. “ShapeNet: A Shapelet-Neural Network Approach for Multivariate Time Series Classification”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 35.9 (May 2021), pp. 8375–8383. DOI: 10.1609/aaai.v35i9.17018. URL: https://ojs.aaai.org/index.php/AAAI/article/view/17018.

[72] Hegui Zhu et al. “TCRAN: Multivariate time series classification using residual channel attention networks with time correction”. In: *Applied Soft Computing* 114 (2022), p. 108117. DOI: https://doi.org/10.1016/j.asoc.2021.108117. URL: https://www.sciencedirect.com/science/article/pii/S1568494621009996.

[73] Rongjun Chen et al. “DA-Net: Dual-attention network for multivariate time series classification”. In: *Information Sciences* 610 (2022), pp. 472–487. DOI: https://doi.org/10.1016/j.ins.2022.07.178. URL: https://www.sciencedirect.com/science/article/pii/S0020025522008738.

[74] Karen Simonyan et al. “Very Deep Convolutional Networks for Large-Scale Image Recognition”. In: (2014). DOI: 10.48550/ARXIV.1409.1556. URL: https://arxiv.org/abs/1409.1556.

[75] Fazle Karim et al. “LSTM Fully Convolutional Networks for Time Series Classification”. In: *IEEE Access* 6 (2018), pp. 1662–1669. DOI: 10.1109/access.2017.2779939. URL: https://doi.org/10.1109%2Faccess.2017.2779939.

[76] Davide Castelvecchi. “Can we open the black box of AI?” In: *Nature* 538 (Oct. 2016), pp. 20–23. DOI: 10.1038/538020a.

[77] Filip Karlo Dosilovic et al. “Explainable artificial intelligence: A survey”. In: *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)* (2018), pp. 0210–0215.

[78] Alejandro Barredo Arrieta et al. *Explainable Artificial Intelligence (XAI): Concepts, Taxonomies, Opportunities and Challenges toward Responsible AI*. 2019. DOI: 10.48550/ARXIV.1910.10045. URL: https://arxiv.org/abs/1910.10045.

[79] Bolei Zhou et al. *Learning Deep Features for Discriminative Localization*. 2015. DOI: 10.48550/ARXIV.1512.04150. URL: https://arxiv.org/abs/1512.04150.

[80] Haofan Wang et al. *Score-CAM: Score-Weighted Visual Explanations for Convolutional Neural Networks*. 2019. DOI: 10.48550/ARXIV.1910.01279. URL: https://arxiv.org/abs/1910.01279.

[81] Ramprasaath R. Selvaraju et al. “Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization”. In: *International Journal of Computer Vision* 128.2 (Oct. 2019), pp. 336–359. DOI: 10.1007/s11263-019-01228-7. URL: https://doi.org/10.1007%5C%2Fs11263-019-01228-7.

[82] Aditya Chattopadhay et al. “Grad-CAM: Generalized Gradient-Based Visual Explanations for Deep Convolutional Networks”. In: *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, Mar. 2018. DOI: 10.1109/wacv.2018.00097. URL: https://doi.org/10.1109%5C%2Fwacv.2018.00097.

[83] Ruigang Fu et al. *Axiom-based Grad-CAM: Towards Accurate Visualization and Explanation of CNNs*. 2020. DOI: 10.48550/ARXIV.2008.02312. URL: https://arxiv.org/abs/2008.02312.

[84] Wojciech Samek et al. *Evaluating the visualization of what a Deep Neural Network has learned*. 2015. DOI: 10.48550/ARXIV.1509.06321. URL: https://arxiv.org/abs/1509.06321.

[85] Andrej Karpathy et al. *Visualizing and Understanding Recurrent Networks*. 2015. DOI: 10.48550/ARXIV.1506.02078. URL: https://arxiv.org/abs/1506.02078.

[86] Dzmitry Bahdanau et al. *Neural Machine Translation by Jointly Learning to Align and Translate*. 2014. DOI: 10.48550/ARXIV.1409.0473. URL: https://arxiv.org/abs/1409.0473.

[87] Marco Tulio Ribeiro et al. *"Why Should I Trust You?": Explaining the Predictions of Any Classifier*. 2016. DOI: 10.48550/ARXIV.1602.04938. URL: https://arxiv.org/abs/1602.04938.

[88] S. Barendswaard et al. "Human Crossfeed in Dual-Axis Manual Control with Motion Feedback". In: vol. 49. Aug. 2016. DOI: 10.1016/j.ifacol.2016.10.514.

[89] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[90] Dominic Masters et al. "Revisiting Small Batch Training for Deep Neural Networks". In: *CoRR* abs/1804.07612 (2018). arXiv: 1804.07612. URL: http://arxiv.org/abs/1804.07612.

[91] Diederik P. Kingma et al. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG].

[92] François-Guillaume Fernandez. *TorchCAM: class activation explorer*. https://github.com/frgfm/torch-cam. Mar. 2020.

[93] Sunitha Basodi et al. "Gradient amplification: An efficient way to train deep neural networks". In: *Big Data Mining and Analytics* 3.3 (2020), pp. 196–207. DOI: 10.26599/BDMA.2020.9020004.

[94] Jacob Gildenblat et al. *PyTorch library for CAM methods*. https://github.com/jacobgil/pytorch-grad-cam. 2021.

Part III

Appendices to Scientific Article

A

Model-Agnostic Hyperparameters

Regarding the model-agnostic hyperparameters, there exists multiple and are as follows:

- Epoch: The number of times the NN model goes through the entire dataset when training.
- Batch Size: The number of training examples utilized in each epoch.
- Loss Function: Used to quantify the discrepancy between the predicted outputs of the NN and the actual values. Minimizing the value from this function is the goal of DL [35].
- Optimiser Function: Used to adjust the model's parameters with the goal of minimizing the loss function.
- Learning Rate: The step size at which the optimizer function adjusts the model's parameters during training.
- Output Activation Function: Used after the final layer of the NN to transform its final predictions into an appropriate format. In this case, it is used to create probabilities of the predicted output classes.
- Number of Kernels: This value will dictate the number of feature maps that the model will produce at the very end of each analyzing convolutional layer, i.e., the three cubes found in the architecture of the RN-XCM model. As the two of the blocks are residual blocks and contain three convolutional layers, this value will apply to the last convolutional layer of the said residual block, $F_{r,\,3}$, with the number of kernels of the two preceding convolutional layers, $F_{r,\,1}$ and $F_{r,\,2}$, being determined based on the ratio of $F_{r,\,1}$, $F_{r,\,2}$, $F_{r,\,3}$.

As these values are model agnostic, the values used will be those that have been used by the majority of the models that were designed for MTS classification [43, 44, 45, 46] alongside ResNet [41]. These values are shown in Table A.1.

Table A.1: The model-agnostic hyperparameter values or methods

Hyperparameter	Values or Methods
Epoch	20
Batch Size	128
Loss Function	Categorical Cross Entropy
Optimiser Function	Adam [91]
Learning Rate	$1e-3$
Output Activation Function	Softmax

As was stated in the paper, the number of kernels was not set to allow for the possibility of better generalization performance out of the RN-XCM model.

B

Average Model Output of MJ Data Compared to de Jong's Numbers

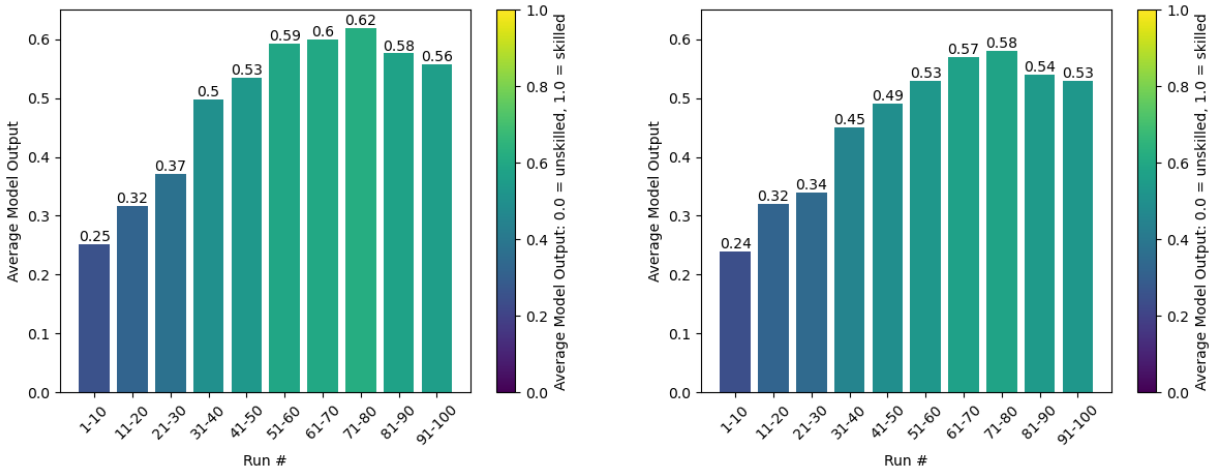

Figure B.1: Average model output of "realistic" scenario of the MJ Data with an increasing number of runs binned. The left figure is the output of the RN-XCM model and the right figure is the output of the one produced by De Jong using ResNet [58].

As can be seen, the outputs from the RN-XCM model are extremely similar to what was obtained from de Jong's study using ResNet.

C

Class Activation Mapping

This paper explored the possibility of adding an explainability aspect to the RN-XCM model. To do so, three different Class Activation Mapping (CAM) [79] models were utilized, Gradient-weighted Class Activation Mapping (Grad-CAM) [81], Axiom-based Grad-CAM (XGrad-CAM) [83], and Grad-CAM++ [82]. The explanations obtained would be spatial-temporal as they can help understand which variables during which time segments were most responsible for a given classification. Furthermore, this approach can show which time segments had the dominant joint contribution to that same classification. The differences between these three CAM methods are as follows:

- Grad-CAM: The original CAM method utilizes the gradients of the target classification with respect to the feature maps of the target convolutional layer to determine which features were most important for the classification [81].
- XGrad-CAM: A variation of Grad-CAM that utilizes gradient maps instead of gradients to determine which features were most important for the classification [83].
- Grad-CAM++: A variation of Grad-CAM that utilizes second-order gradients instead of first-order gradients to determine which features were most important for the classification [82].

These CAM methods were used to determine which variables at which time steps were considered most important by the RN-XCM model for classifying a sample as either ”skilled” or ”unskilled” in Phase 1. This step was done only for Phase 1, i.e. the use of the RW Data, as the focus of this study is the application of NNs for multi-axis tracking tasks. In addition, this step was done solely for the RN-XCM and not the ResNet model as in its original form, ResNet is incompatible with the Python package used to obtain the CAMs [94]. In order to make it compatible, modifications to the ResNet model would be required which would compromise the fundamental design choices underlying the original model architecture. Therefore, any obtained CAMs would not be an accurate representation of the original NN model’s analysis methods. Therefore, obtaining the CAMs for ResNet was not performed.

The results of obtaining the CAMs of the RN-XCM model for the complete RW Data in the ”realistic” scenario are presented in Figure C.1 and Figure C.2 which shows the variables-wise CAMs and the latter presents the temporal-wise CAMs respectively. The former details which input variables during which time segments were most responsible for a given skill classification and the latter details which time segments were the joint contribution of all the input variables were most important for the same classification.

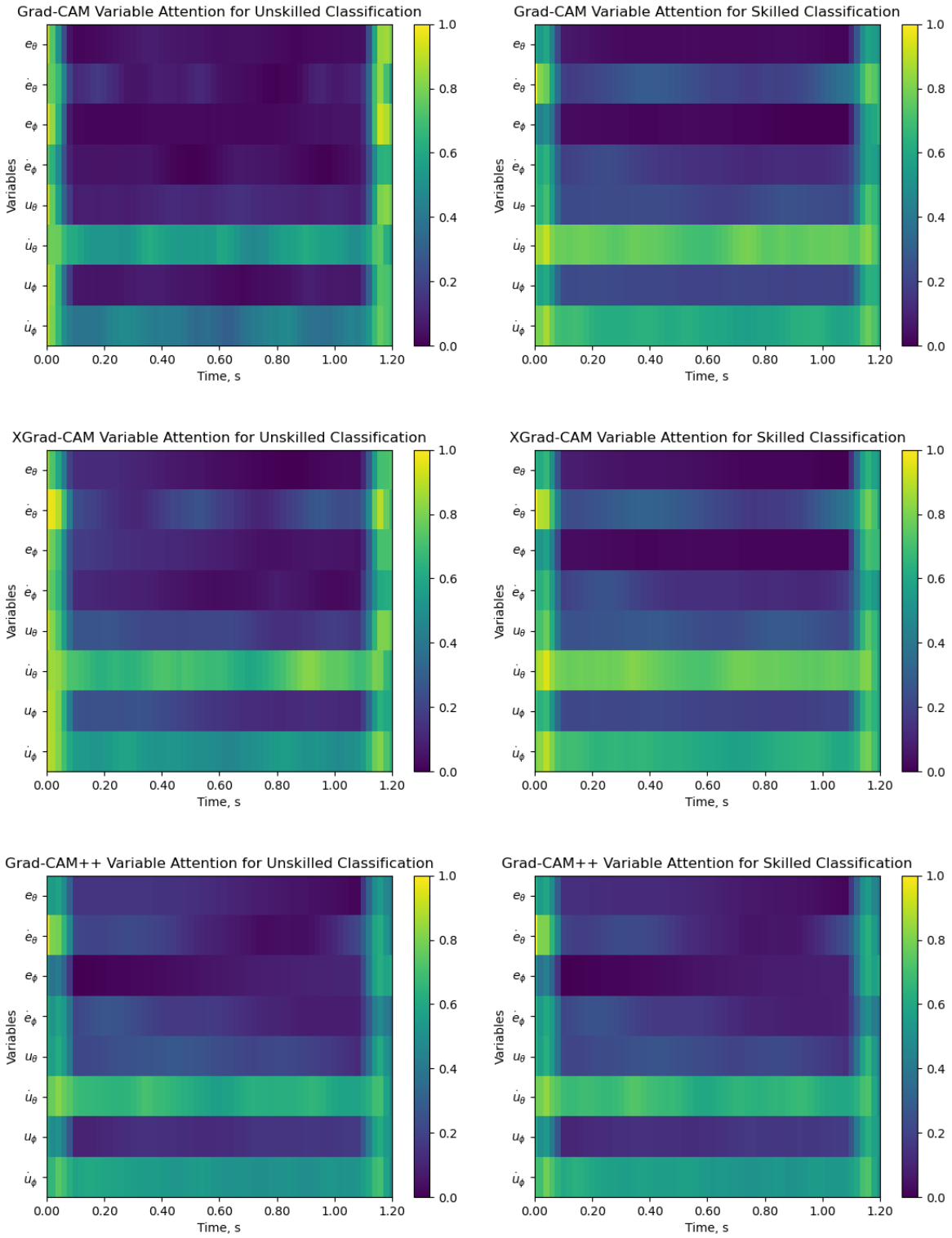

Figure C.1: Global average of all variable-wise CAMs of RN-XCM when making either unskilled or skilled predictions for the complete RW Data in the ”realistic” scenario.

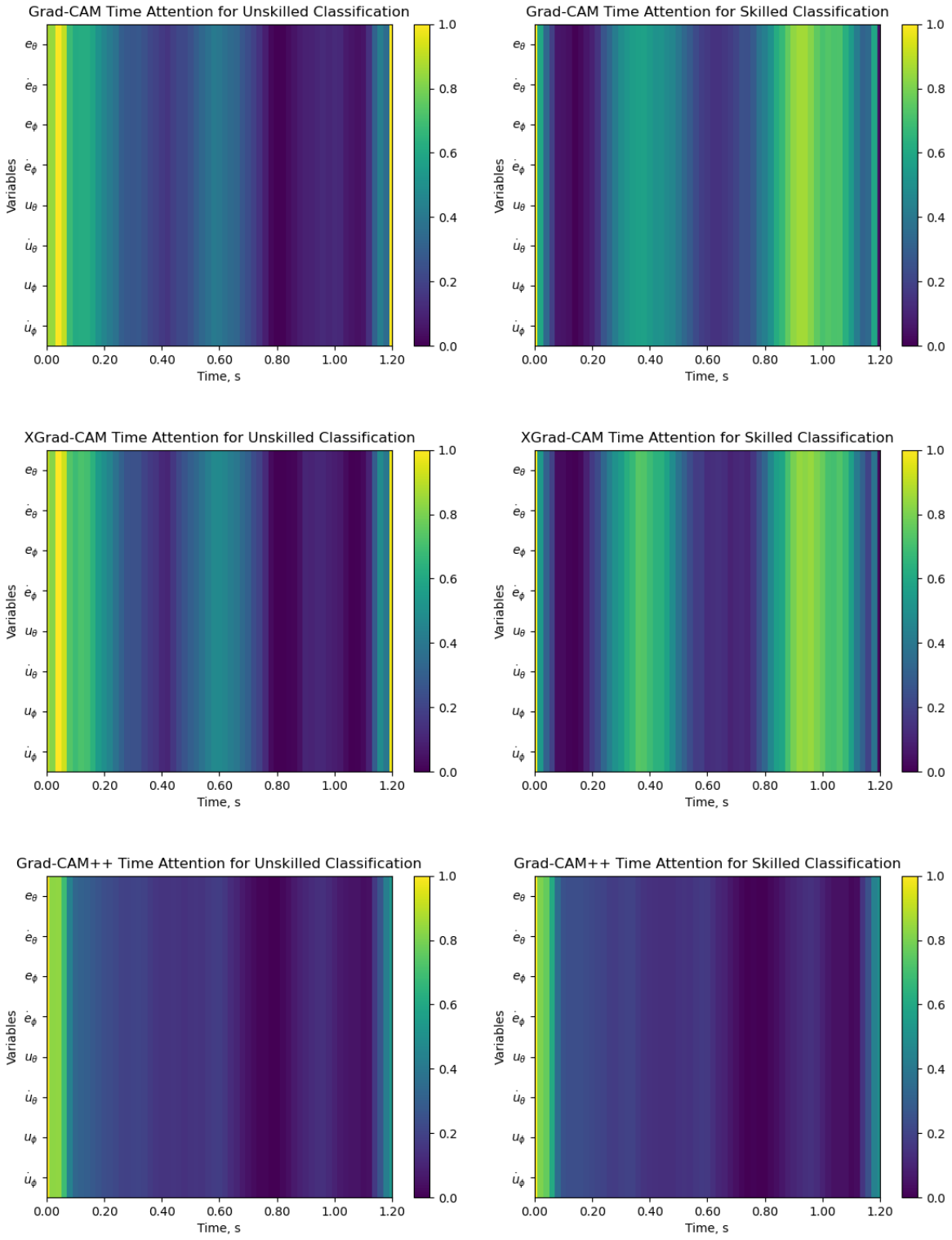

Figure C.2: Global average of all temporal-wise CAMs of RN-XCM when making either unskilled or skilled predictions for the complete RW Data in the ”realistic” scenario.

Overall, when looking at all of the CAMs, a few general observations can be made. The CAMs produced by Grad-CAM and XGrad-CAM are exceptionally similar to one another alongside a trend of looking at different regions between making ”unskilled” and ”skilled” classifications. In contrast, the CAMs obtained by

Grad-CAM++ are more often than not distinct from the CAMs obtained from the other two methods but look at the same regions regardless of making ”unskilled” and ”skilled” classifications. This is predominantly caused by the fact that the Grad-CAM and XGrad-CAM methods utilize first-order gradients to produce their CAMs [81, 83] whereas Grad-CAM++ utilizes second-order gradients to produce its CAMs [82].

Starting with the variable-wise CAMs, it can be seen that overall, the derivative of the HO input signals in both axes, $\dot{u}_\theta$ and $\dot{u}_\phi$, was the two most important signals with the rest of the signals having relatively lower importance. To be specific, when looking at the importance of each variable in creating a classification for both scenarios, as seen in Figure C.3, these signals each only contributed an average of 11.51% to the model's output compared to the aforementioned $\dot{u}_\theta$ and $\dot{u}_\phi$ average individual contribution of 17.04%. A significant trend that can be seen with the variable-wise CAMs was the model's preference for the use of the first-order derivative signals compared to their original raw signals. For the ”ideal” scenario, the model placed 16.28% more importance on the first-order derivative signals, and for the ”realistic” scenario, the model placed 13.50% more importance on the first-order derivative signals. This signifies that these derivative signals contain certain characteristics that make them more desirable and reliable for the NN model when making their skill-level classifications.

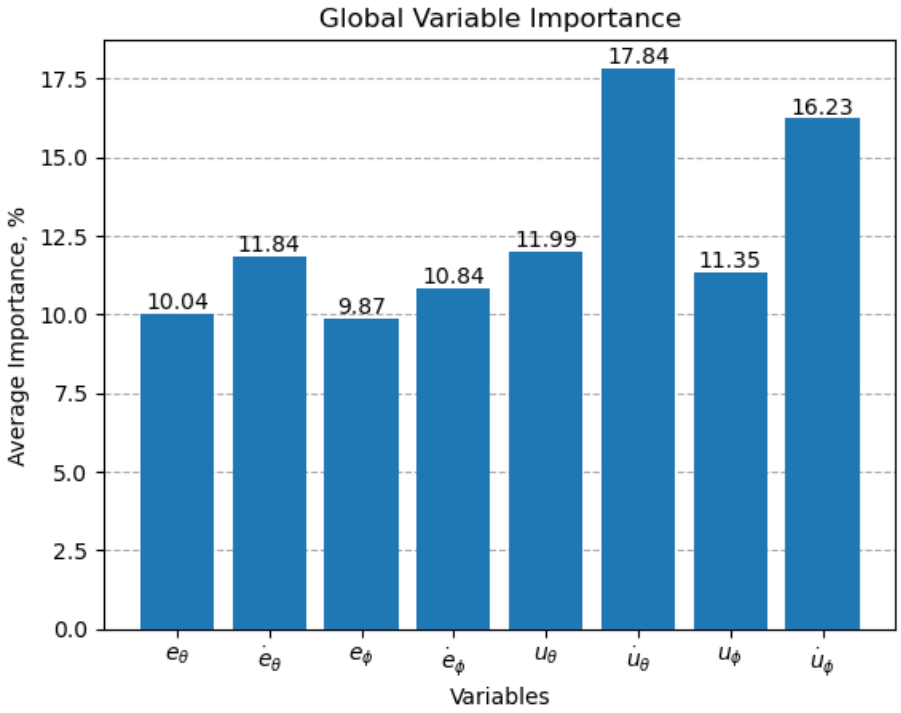

Figure C.3: Average global variable importance of all CAM methods for both classifications in the ”realistic” scenario for the complete RW Data.

Regarding the temporal-wise CAMs, it can be seen overall that they are not as bright as the variable-wise CAMs. This signifies that the model looks at the individual contributions of the signals more often than the joint contribution of all the signals. However, unlike variable-wise CAMs, the interpretability of temporal-wise CAMs is notably limited, thus raising questions about their practicality within the context of explainability and feedback.

In general, these explainability methods produced significantly different explanations from those obtained by De Jong [58] as their explainability methods incorporated model-agnostic methods such as SHAP [66] whereas this study incorporated model-specific methods. Additionally, when delving further into the observed preference of the model to use the first-order derivative signals over the regular signals in both scenarios, this could be caused by numerous factors that only the first-order derivative signals can capture. The control task from which the RW Data originated involves continuous adjustments to the control inputs, and therefore a continuous change to the error signal as well. The first-order derivative then captures the temporal dynamics of the HO control behavior as by definition, it represents the rate of change of

the signals over time. To be specific, the use of the derivative signals implicitly incorporates a level of feature engineering as it highlights inflection points and sharp changes from the original signal, acting as informative features for the NN model. In other words, the first-order derivative acts as a high-pass filter as it highlights rapid and subtle variations in the control signal, which are indicative of skillful control. Therefore, the derivative signals allow for a more discerning differentiation between ”skilled” and ”unskilled” data for the NN model.

Overall, the use of Gradient-Based CAM methods would allow for more detailed real-time feedback on pilot control behavior which would be substantial for not only real-time usage but also for simulator training of inexperienced pilots. However, to do so, the features used to allow for skill classification should be limited to those that the HO is able to directly influence, i.e. the raw input signal u. The usage of the error signals e and first-order derivative signals would cause interpretation and debriefing for skill improvement to be more difficult. Alongside this, it would also be advised that more features should be included such as throttle, rudder pedals, elevator control, and so forth as it would allow for more robust feedback and potentially more accurate skill level classifications as an increase in input features would help the NN to learn more complex relationships and patterns that would be missed with fewer features. In addition, the presence of temporal-wise CAMs adds further difficulty in interpretation, and therefore any research that aims to develop an interpretable real-time feedback model would benefit from a removal of the temporal analyzing aspect of the RN-XCM architecture. This however has the potential to lower the classification performance of said model. Finally, the addition of explainability allowed by the use of the Grad-CAM methods would allow for various other applications in high-stakes domains such as aviation as regulators may mandate models to be interpretable and provide explanations to ensure trust, safety, and compliance with relevant standards.

D

Real Time Standardization

In order to create a true real-time skill level classifying NN model, the standardizing of control per window rather than per run as was done within the paper was explored. Two different standardization methods per window were evaluated against the used standardization method per run. These two methods are as follows:

- Normalization based on the largest values of each signal within each window.

$$\hat{u}(t) = \frac{\hat{u}(t)}{max(|min(\overrightarrow{\hat{u}})|, |max(\overrightarrow{\hat{u}})|)} \tag{D.1}$$

- Normalization based on the magnitude of each signal within each window.

$$\hat{u}(t) = \frac{\hat{u}(t)}{max(||\hat{u}(t)||)} \tag{D.2}$$

When these standardization methods were used to obtain accuracies for the ideal scenario for the RW data as was done with standardization methods performed per tracking run, the results obtained were as follows:

Table D.1: Ideal Accuracies Obtained by RN-XCM per Standardization Method

Standardization Method	Ideal Accuracies, %
Per Window Normalization based on Largest Values	50.00
Per Window Normalization based on Magnitude	50.00
Per Run Standardization	93.50

As can be seen in Table D.1, the normalization methods per window result in extremely poor performance compared to the per-run standardization. In fact, the obtained accuracies of 50 % imply that the model classifies every window of tracking data it sees as one skill level, therefore being unable to discriminate the skill levels of the tracking data.